Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Кафедра автоматики та управління в технічних системах
(повна назва кафедри)

До захисту допущено
Завідувач кафедри
_____ Ролік О.І.__
(підпис) (ініціали, прізвище)
“___”_____2019_р.

Дипломний проект

на здобуття освітнього ступеня “бакалавр”
(назва ОС)

за напрямом 6.050201 “Системна інженерія”
(код та назва напряму підготовки)

на тему: Система рекомендації лотів аукціону

Виконав: студент __4__ курсу, групи ____ІА-52_____
(шифр групи)

_____Парфенюк Тарас Миколайович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____аспірант Дорога-Іванюк О.О._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант_____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019_ року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних
(повна назва)

Освітній ступінь бакалавр
Напрям підготовки 6.050201 "Системна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Ролік О.І. _____
(прізвище ініціали) (підпис)
"___"_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Парфенюк Тарас Миколайович
(прізвище, ім'я, по батькові)

1. Тема проекту: Система рекомендації лотів аукціону,
Керівник проекту: Дорога-Іванюк Олена Олександрівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _06__ квітня __2019___ р. №___-с

2. Термін подання студентом проекту 15 червня 2019 р.
3. Вихідні дані до проекту
База даних переглядів та покупок користувачів комерційної платформи .
4. Зміст проекту
Дослідження методів надання рекомендацій та побудова моделі для прогнозування покупок користувача аукціону.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 5 березня 2019 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на дипломний проект	06.03.2019 – 16.03.2019	Виконано
2	Аналіз предметної області та вимог завдання, огляд існуючих рішень	17.03.2019 – 29.03.2019	Виконано
3	Аналіз даних історії переглядів та покупок	30.03.2019 – 15.04.2019	Виконано
4	Попередня обробка даних лотів та історії переглядів користувачів	16.04.2019 – 28.04.2019	Виконано
5	Дослідження вимог до моделі та обмежень, вибір методу генерації рекомендацій	29.04.2019 – 14.05.2019	Виконано
6	Побудова моделі прогнозування покупок користувача на базі графу суміжних переглядів	16.04.2019 – 31.05.2019	Виконано

7	Підготовка матеріалів до друку та оформлення пояснювальної записки	01.06.2019 – 09.06.2019	Виконано
8	Підготовка доповіді до захисту та оформлення ілюстративного матеріалу	10.06.2019 – 18.06.2019	Виконано

Студент _____ **Парфенюк Т. М.**
(підпис) (прізвище та ініціали)

Керівник проекту _____ **Дорога-Іванюк О.О**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Парфенюк Т.М. Система рекомендації лотів аукціону. КПІ ім. Ігоря Сікорського, Київ, 2019.

Проект містить: 75 с., 1 табл., 28 рис., 21 посилання на джерела.

Ключові слова: рекомендаційна система, аукціон, графи.

Об’єктом розробки є система рекомендації лотів аукціону.

Мета розробки – побудова моделі для прогнозування покупок користувача аукціону та надання рекомендацій, опираючись на історію переглядів та покупок.

У дипломному проекті розроблено систему рекомендації лотів аукціону. У ході роботи над проектом було розглянуто основні методи створення рекомендацій та можливість їх використання в рамках інтернет-аукціону. За допомогою алгоритмів кластерного аналізу та статистичних методів було побудовано модель на основі графу для прогнозування покупок користувача та генерації рекомендацій, що опирається на історію переглядів та покупок.

Побудована модель дозволяє полегшити процес пошуку лотів для користувача та збільшити його лояльність до платформи з першого відвідування, що сприяє зростанню кількості проданих лотів аукціону.

SUMMARY

Parfeniuk T.M. Recommendation system for auction lots. Igor Sikorsky KPI, Kyiv, 2019.

The project contains 75 p., 1 table, 28 figures, 21 references to the source.

Keywords: recommendation system, auction, graphs.

The object of development is recommendation system for auction lots. The purpose of the development is to develop a model for forecasting auction user purchases and providing recommendations, based on the history of views and purchases.

The graduation project has developed a system of recommendations for auction lots. The main methods of creating recommendations and the possibility of their use in the framework of the Internet auction were considered during the work on the project. A graph-based model for predicting user purchases and generating recommendations based on the history of views and purchases was developed, using cluster analysis algorithms and statistical methods.

The developed model helps to simplify the lots search process for the auctions customers and increase their loyalty to the platform from the first visit, which contributes to the increase in the number of auctioned lots sold

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

C2C	Customer to customer;
DBSCAN	Density-based spatial clustering of applications with noise;
IBM	International Business Machines Corporation;
SQL	Structured query language;
CSV	Comma-Separated Values;
CQL	Cypher query language;
PCA	Principal component analysis;

ВСТУП

В Інтернеті на даний момент налічуються колосальні об’єми різноманітного контенту, які постійно ростуть. З ростом кількості інформації, що є доступною для користувача, зростає актуальність методів пропонування користувачеві потенційно цікавої йому інформації, адже пошук потрібної інформації звичними способами стає нетривіальною задачею. Саме тому з середини 90-х років виникли й почали розвиватися рекомендаційні системи. Використання рекомендаційних систем покликано спростити взаємодію користувача та веб-сайту. Такі системи формують рекомендації для кожного користувача базуючись на його попередніх діях, придбаних товарах, залишених відгуках, тощо.

В наш час рекомендаційні системи використовуються багатьма комерційними та соціальними веб-сайтами. Прикладами є такі гіганти як YouTube, eBay, Amazon, Netflix. З українських сервісів можна виділити інтернет-магазин Rozetka, що також використовує механізми рекомендацій.

Інтернет-аукціони є популярним засобом для купівлі та продажу товарів різних областей. У формі аукціону здебільшого здійснюється торгівля товарами або послугами, що мають певні індивідуальні характеристики. Таким чином унеможливлюється постійне забезпечення ринку партіями того чи іншого товару. До того ж кожен лот має обмежений час, коли він є доступний для огляду або торгів. Враховуючи дані умови, рекомендаційна система аукціону повинна створювати релевантні рекомендації не спираючись на конкретні лоти, а використовуючи опис лоту. Не менш важливою вимогою в умовах динамічної природи каталогу лотів аукціону є здатність системи генерувати рекомендації за якомога менший час.

Сучасні веб-сервіси мають можливість збирати значні об’єми інформації про поведінку користувачів, їхні вподобання та інше. Наявність великої кількості даних дозволяє будувати доволі точні моделі для прогнозування

поведінки користувачів та створення релевантних рекомендацій. При побудові таких моделей потрібно звертати особливу увагу на такі завдання як дотримання конфіденційності даних користувача та протидія проявам нечесної конкуренції, адже від результатів їх вирішення залежить лояльність користувачів до даної системи надання рекомендацій.

Актуальність проекту полягає у тому, що зі постійним збільшенням обсягів даних у мережі інтернет, комерційним платформам потрібні засоби для впорядкування цих даних та полегшення роботи з ними кінцевого споживача. Для таких засобів електронної комерції як онлайн магазини та аукціони важливо, щоб користувач міг швидко знайти потрібний йому товар та в результаті придбати його. Використання рекомендаційної системи може зменшити час, що витрачається на пошуки, а також сприяти виявленню нових товарів, що можуть бути потенційно цікавими для користувача. Таким чином, в поле зору користувача потрапляє більша кількість товарів, що сприяє збільшенню кількості продажів. Саме тому компанії-власники подібних ресурсів зацікавленні у впровадженні та розвитку подібних систем.

Об’єктом дослідження є інтернет аукціони з товарами порівняно високої вартості, на кшталт аукціону б/у автомобілів copart.com.

Метою проекту є розробка рекомендаційної системи для інтернет аукціону на прикладі аукціону б/у автомобілів, що при генерації рекомендацій спиратиметься на історію активності користувачів, їх попередні покупки та характеристики лотів, а також зможе працювати в умовах динамічного списку товарів. Загальний аналіз та порівняння методів машинного навчання, їх застосування в алгоритмах надання рекомендацій.

1 ПОСТАНОВКА ЗАДАЧІ

Метою дипломної роботи є дослідження та розробка системи рекомендації лотів аукціону. Основним завданням рекомендаційної системи є надання користувачеві релевантних рекомендацій, враховуючи при цьому його інтереси. Також на систему накладаються додаткові вимоги, враховуючи особливості предметної області. Отже, проектована система повинна:

— Враховувати поведінку користувача;

— Ефективно працювати незалежно від кількості існуючих товарів;

— Ефективно працювати незалежно від кількості користувачів;

— Створювати рекомендації в умовах динамічного каталогу товарів;

Для досягнення даної цілі було вирішено наступні завдання:

— Дослідження проблем, пов’язаних зі створенням рекомендацій в контексті аукціону.

— Проведення огляду існуючих методів рекомендацій.

— Розробка рекомендаційної системи, яка створює перелік пропонованих лотів, базуючись на характеристиках лотів та історії переглядів та покупок користувачів.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Вступ

В даному розділі буде розглянуто основні теоретичні відомості про предметну область, а саме поняття про інтернет-аукціон та його особливості у порівнянні зі звичайними комерційними онлайн-платформами. Також буде проведено ознайомлення з поняттям про рекомендаційні системи, розглянуто базові структуру рекомендаційної системи та методи створення рекомендацій.

Також в межах розділу приведено теоретичні відомості про засоби, використовувані при реалізації рекомендаційних систем, а саме огляд методів та інструментів кластерного аналізу та базових знань про статистику та теорію ймовірностей.

2.2 Рекомендаційні системи

У сучасному світі рекомендаційні сервіси стали настільки поширеними, що ми просто не помічаємо як вони полегшують нам робити вибір при здійсненні різноманітних покупок. На даний момент майже будь-яка комерційна платформа або сервіс має свою систему надання рекомендацій.

Рекомендаційна система, це механізм, що намагається передбачити інтерес користувача та запропонувати йому найбільш підходящий для нього товар або медіа ресурс. Однією з переваг, що надаються бізнесу при використанні рекомендаційної системи є здатність такої системи допомогти користувачам відкривати для себе нові товари, або ж медіа контент. Це добре, коли користувач знаходить те, що він шукає, або навіть більше. Останній пункт має безпосереднє значення для бізнесу, тому що відкриття нового контенту залишає шанс для продовження взаємодії користувача з ресурсом. Саме такі рекомендаційні системи приносять вагому користь для бізнесу.

Не менш важливим є те, що система надання рекомендацій дозволяє зробити взаємодію з ресурсом приємнішою та простішою, адже

рекомендаційна система бере на себе відсіювання не цікавої для користувача інформації. А так як користувач витрачає менше часу на пошуки, він може більше зосередитись на споживанні.

В наш час системи надання рекомендацій, є одним з наріжних каменів обслуговування клієнтів. Вони значно полегшують досягнення ідеї «надавати користувачеві те, що йому потрібно». Крім того, ці системи є гарним способом вивчення власної цільової аудиторії та її потреб. Таким чином, рекомендаційні системи є незамінною частиною сучасної комерції.

2.2.1 Базова структура рекомендаційної системи

Основними завданнями, що стоять перед рекомендаційною системою є збільшення кількості проданих товарів або послуг, підвищення лояльності користувача до сервісу та покращення розуміння бажань користувачів надавачем послуг [3]. Для вирішення цих завдань при розробці рекомендаційних систем використовують такі базові підходи як колаборацийна фільтрація, фільтрація на базі вмісту, або їх поєднання [4].

Для рекомендаційних систем типовим є виділення двох основних таких етапів створення рекомендацій:

Відбір кандидатів (Candidate generation)

Відбір кандидатів є першим та основним кроком у побудові рекомендаційної вибірки. Метою даного етапу є виділення відносно невеликої порції даних, які були б потенційно цікавими для користувача [5]. Методи, що використовуються на даному етапі буде розглянуто в розділах 2.2, 2.3 та 2.4.

— Впорядкування (Scoring)

Полягає у впорядкуванні попередньо отриманого набору даних та вибору з нього документів для демонстрації користувачеві [6].Після чого системою враховуються додаткові вимоги до впорядкованої вибірки, наприклад в контексті аукціону можуть відсіюватися лоти, в торгах за які користувач вже брав участь, тощо.

2.2.2 Типи фільтрації

2.2.3 Колаборaційна фільтрація

Колаборaційна фільтрація дозволяє створювати рекомендації, що ґрунтуються на моделі поведінки користувачів. Модель може бути сконструйовано як з урахуванням поведінки інших користувачів зі схожими рисами, або ж використовуючи дані про поведінку лише одного користувача. Метод колаборaційної фільтрації враховує інформацію про вподобання тої чи іншої групи користувачів, тобто користувачеві пропонуватимуться товари, що вже були вподобані іншими користувачами. Для реалізації даного методу використовуються алгоритми обчислення схожості між користувачами, для групування за поведінкою та іншими критеріями. При використанні даного методу особливу увагу необхідно виділити проблемі конфіденційності, так як метод використовує історію активності користувачів, що в свою чергу дозволяє відтворити портрет конкретного користувача, спираючись на інформацію про надані йому рекомендації.

При реалізації рекомендаційної системи з на базі колаборaційної фільтрації в свою чергу можна виділити алгоритми на основі пам’яті та алгоритми на основі моделі. Їх відмінність полягає в методах роботи з даними [7].

Для роботи алгоритмам на основі пам’яті потрібно вивантажувати в пам’ять інформацію про елементи та користувачів, для побудови рекомендацій в момент запиту. Такий підхід має суттєві недоліки, у вигляді низької швидкодії, значне споживання пам’яті та як результат –погана масштабованість.

Алгоритми на основі моделі не мають таких недоліків і полягають в побудові моделі, що дозволяє розрахувати рекомендації. На відміну від алгоритмів на основі пам’яті, така модель може розраховуватись оф лайн. Для можливості працювати з новими даними, модель повинна оновлюватись

відповідно до змін. Для оптимізації даного процесу нові дані можуть збиратись в групи, після чого модель перераховується з урахуванням групи даних.

Серед переваг такого підходу можна виділити відсутність потреби в досконалому вивченні предметної області. Також позитивним є можливість моделі допомагати користувачам виявляти нові інтереси.

Недоліками є проблема конфіденційності користувачів, а також проблема холодного старту, так як для рекомендації нового елементу система повинна зібрати достатню кількість оцінок користувачів [8].

Також суттєвим недоліком традиційних реалізацій даного методу фільтрації є те, що обчислення або зовсім не виконуються, або виконуються лише частково оф лайн. Алгоритм не є практичним для великих наборів даних, за відсутності використання зменшення розмірності, вибірки або розділення даних, що в свою чергу приводить до зменшення релевантності наданих рекомендацій.

2.2.4 Колаборaційна фільтрація на базі кластерних моделей

Для пошуку користувачів зі схожими інтересами, кластерні моделі поділяють користувачів на групи, базуючись на їх поведінці. Подальша робота алгоритму зводиться до завдання класифікації. Метою алгоритму є призначення користувачеві групи, що містить користувачів з найбільш схожих схожою поведінкою та відповідно, інтересами.

Для розподілу користувачів по групах, алгоритмом використовуються метрики схожості. Поширеним методом обчислення подібності є розрахунок косинусу кута між векторними представленнями профілів користувачів (2.1):

$$S(\vec{A},\vec{B}) = cos(\vec{A},\vec{B}) = \frac{\vec{A}\cdot\vec{B}}{\|\vec{A}\|\cdot\|\vec{B}\|}, \quad (2.1)$$

де $\vec{A}, \vec{B}$ – вектори, що є представленнями користувачів та мають M вимірів, що відповідають товарам, які були ними оцінені.

До недоліків відноситься необхідність у великих вибірках даних для ефективної роботи моделі, а також низька якість рекомендацій для великих кластерів.

Серед переваг відзначається краща масштабованість та продуктивність, ніж у класичних алгоритмів колаборaційної фільтрація, оскільки вони порівнюють користувача з не з усією базою користувачів, а лише з обмеженим числом кластерів.

2.2.5 Фільтрація на базі вмісту

Фільтрація на базі вмісту враховує характеристики об’єктів для рекомендації користувачеві об’єктів, подібних до раніше вподобаних, використовуючи функцію відстані.

Цей підхід полягає в створенні профілю користувача для чого системою враховуються оцінка елементів конкретним користувачем. Також системою формуються представлення елементів у вигляді набору характеристик, для подальшого обчислення схожості між елементами [9]. Для цього використовують такі методи машинного навчання, як байєсівський класифікатор, алгоритми кластеризації, дерева рішень та штучні нейронні мережі [10].

До переваг даного підходу можна віднести те, що модель не потребує даних про інших користувачів, оскільки рекомендації є специфічними для конкретної особи. Це полегшує масштабування при значній кількості користувачів. Також модель може враховувати конкретні інтереси користувача.

Серед недоліків відзначається потреба в детальному розумінні предметної області для побудови представлення елементу в системі та вибору

функції відстані. Також модель може створювати рекомендації лише на основі існуючих інтересів користувача.

2.2.6 Оцінка якості наданих рекомендацій

Звичайні методи оцінки якості як точність прогнозування, що можуть бути застосовані на етапі розробки системи не є ефективними в контексті рекомендаційних систем. Враховуючи специфіку роботи таких систем, а саме взаємодія з інтересами користувачів та їх прогнозування, найкращим способом оцінки якості роботи системи є показ користувачеві рекомендацій, та спостереження його реакції на пропоновані йому товари або медіа-контент. Тим не менш можна виділити метрики, що відображають ефективність роботи системи надання рекомендацій.

До традиційних метрик, що можуть охарактеризувати роботи системи відносяться класичні метрики оцінки прогнозів, а саме:

оцінка середньоквадратичної похибки (RMSE).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{r}_{ui} - r_{ui})^2}{n}}, \quad (2.2)$$

середньоквадратичне відхилення (MSE).

$$MSE = \frac{\sum_{i=1}^{n}(\hat{r}_{ui} - r_{ui})^2}{n}, \quad (2.3)$$

середня абсолютна похибка (MAE).

$$MAE = \frac{\sum_{i=1}^{n}|\hat{r}_{ui} - r_{ui}|}{n}, \quad (2.4)$$

де $\hat{r}_{ui} - r_{ui}$ – різниця між дійсним значенням та спрогнозованим для n випадків.

Додатково для оцінки вводяться також такі поняття як recall та precision.

Recall – здатність моделі знаходити усі релевантні елементи та рекомендувати їх користувачеві.

Precision – співвідношення кількості елементів з високою релевантністю до загальної кількості рекомендованих елементів.

Визначення релевантності елементу визначається відштовхуючись від особливостей предметної області. В якості релевантних елементів можуть розглядаються позитивно оцінені користувачем елементи, наприклад це може бути вподобане відео, або придбаний продукт з позитивним відгуком від даного користувача.

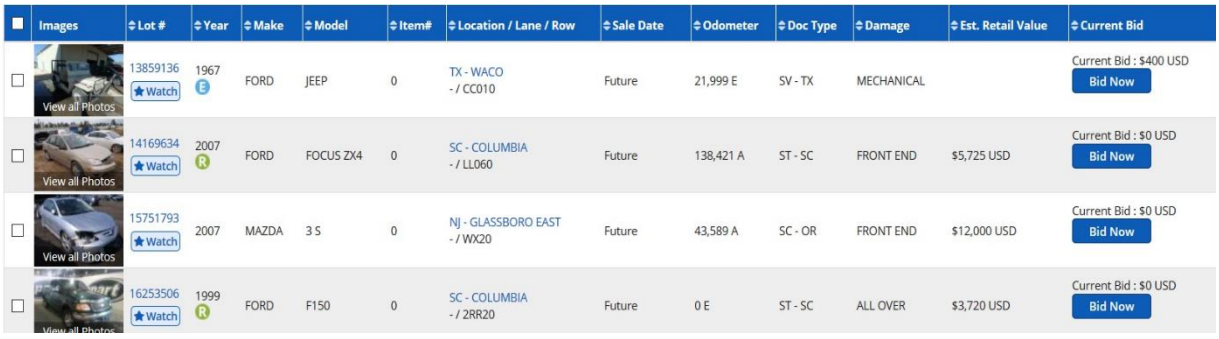
2.3 Онлайн аукціон

За визначенням [1] аукціоном називається механізм купівлі та продажу товарів або послуг, в основі якого лежить конкуренція між учасниками торгу. Даний підхід сприяє підвищенню ціни на товар та максимізації прибутку від його продажу.

До недоліків класичних аукціонів можна віднести складність організації проведення, так як для участі необхідною є особиста присутність учасників. Інтернет аукціони є значно простішими в організації та доступнішими як для учасників, так і для продавців. Можливість брати участь у торгах з будь-якої точки світ та у будь-який час сприяє значному поширенню та популяризації таких аукціонів.

На відміну від інших засобів електронної комерції, як наприклад інтернет магазини, для інтернет аукціонів є поширеним такий підхід надання послуг як consumer-to-consumer [2]. Прикладом сервісу з таким принципом роботи є eBay. Особливостями схеми C2C є низькі трансакційні витрати відносно нижча ціна за товар. Недоліком даної схеми є підвищений ризик шахрайства. Для боротьби з обманом споживачів сервіси вводять системи репутації. Показник репутації продавця повинен враховуватись рекомендаційною системою при рекомендації його товарів.

В даній роботі робитиметься акцент на більш вузько спеціалізовані інтернет аукціони, з порівняно високою середньою вартістю лотів, як наприклад інтернет-аукціон б/у автомобілів Copart. Особливістю таких аукціонів є структура опису лотів, що є постійною. Приклад опису лотів показано на рисунку 2.1. Така особливість дає змогу використовувати засоби фільтрації на базі вмісту.



Images	Lot #	Year	Make	Model	Item#	Location / Lane / Row	Sale Date	Odometer	Doc Type	Damage	Est. Retail Value	Current Bid
View all Photos	13859136 Watch	1967	FORD	JEEP	0	TX - WACO - / CC010	Future	21,999 E	SV - TX	MECHANICAL		Current Bid : $400 USD Bid Now
View all Photos	14169634 Watch	2007	FORD	FOCUS ZX4	0	SC - COLUMBIA - / LL060	Future	138,421 A	ST - SC	FRONT END	$5,725 USD	Current Bid : $0 USD Bid Now
View all Photos	15751793 Watch	2007	MAZDA	3 S	0	NJ - GLASSBORO EAST - / WX20	Future	43,589 A	SC - OR	FRONT END	$12,000 USD	Current Bid : $0 USD Bid Now
View all Photos	16253506 Watch	1999	FORD	F150	0	SC - COLUMBIA - / 2RR20	Future	0 E	ST - SC	ALL OVER	$3,720 USD	Current Bid : $0 USD Bid Now

Рисунок 2.1 – Короткий опису лотів аукціону Copart

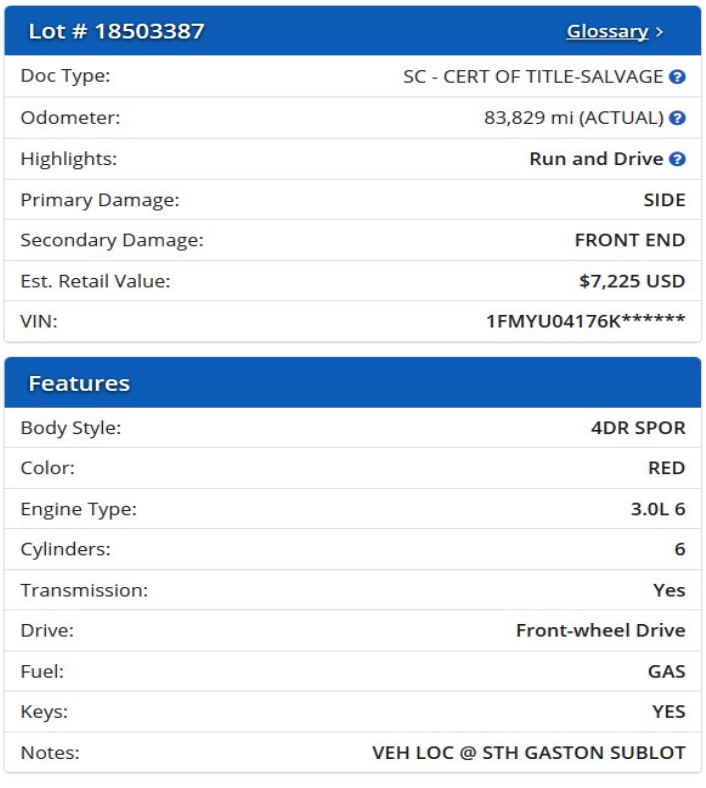
Ще однією особливістю такого типу аукціонів є відносно висока цінність лотів, порівняно з торговими майданчиками загального призначення. Така специфіка сприяє тому, що окремо взятий користувач буде активним лише деякий час, після чого його інтерес до перегляду лотів втрачається. Як результат, з'являється складність для прогнозування інтересів користувача, через неможливість побудови повноцінного профіля. Відповідно на рекомендаційну систему накладаються додаткові вимоги, а саме можливість прогнозування зацікавленості користувача в лоті спираючись на мінімальну кількість даних про попередній його досвід взаємодії з платформою.

2.3.1 Огляд характеристик лоту

Незалежно від конкретного сервісу інтернет аукціону будь-який лот містить інформацію про початкова вартість та продавця. Як було сказано раніше, задля запобігання шахрайству кожен продавець в свою чергу має показник репутації. Даний показник може використовуватись

рекомендаційною системою на етапі впорядкування рекомендаційної вибірки таким чином, що лоти, виставлені користувачами з вищим показником репутації матимуть більшу імовірність потрапити в список рекомендованих.

Також, лот повинен містити інформацію про, власне, товар або послугу, що виставляється на продаж. В більшості інтернет аукціонів опис товару подається у вигляді набору характеристик. Набір характеристик залежить від виду товару, його категорії. Відповідно кожна категорія товарів може мати свій унікальний набір характеристик для опису товару. Важливим для подальшої роботи буде відмітити, що значення даних характеристик може бути подане як у вигляді числа, так і текстового опису. У випадку аукціонів більш вузької спеціалізації всі товари належать до одного типу, відповідно їх структура їх опису є здебільшого незмінною. Для прикладу наведемо вище згаданий Copart. Спеціалізацією даної платформи є б/у автомобілі, що враховується у структурі опису лотів. Відповідно до характеристик лоту



Lot # 18503387	Glossary ›
Doc Type:	SC - CERT OF TITLE-SALVAGE
Odometer:	83,829 mi (ACTUAL)
Highlights:	Run and Drive
Primary Damage:	SIDE
Secondary Damage:	FRONT END
Est. Retail Value:	$7,225 USD
VIN:	1FMYU04176K******

Features	
Body Style:	4DR SPOR
Color:	RED
Engine Type:	3.0L 6
Cylinders:	6
Transmission:	Yes
Drive:	Front-wheel Drive
Fuel:	GAS
Keys:	YES
Notes:	VEH LOC @ STH GASTON SUBLOT

Рисунок 2.2 – Детальний опис лоту
на сайті Copart

віднесено: марка автомобіля, модель, рік виготовлення, місце знаходження лоту, величина пробігу, вид пошкоджень, тип палива, об'єм двигуна і тд. Більш детальний набір характеристик зображено на рисунку 2.2.

2.3.2 Історія активності користувача інтернет аукціону

Для надання більш релевантних рекомендацій, система може використовувати дані про попередні дії користувача на сайті інтернет аукціону. Історія взаємодії користувача може розповісти про звички, інтереси, вподобання та мотивацію користувача у відвідуванні ресурсу.

В рекомендаційних системах відгуки користувачів відіграють важливу роль, так як це очевидний спосіб виявити товар, що сподобався чи навпаки не сподобався покупцеві. Враховуючи специфіку інтернет аукціонів, а саме неможливість оцінити товар після його покупки, в якості неявної оцінки можна використати акт викупу лоту користувачем.

Важливим зауваженням до процесу надання рекомендацій є те, що поведінка користувачів при пошуку та купівлі товарів відрізняється в залежності від типу товару. Наприклад при покупці книг, музики та інших недорогих товарів люди схильні повертатися до платформи задля повторної покупки предмету через короткий проміжок часу. Для товарів порівняно високої вартості така поведінка користувача не є властивою. Наприклад при покупці телевізора, покупці, як правило, розглядають значну кількість позицій, але купують лише один. Проте спільно з дорожчими товарами покупці схильні розглядати товари, що є доповненням до основної, дорожчої покупки. Звернувшись до раніше згаданого прикладу з покупкою телевізора такими допоміжними товарами може бути blu-ray програвач, або ж настінне кріплення.

Особливість полягає в нижчій частоті покупок, що припадають на користувача. Користувачі таких платформ переглядають порівняно велику кількість позицій каталогу перед покупкою, до того ж після придбання такий

користувач може тривалий не відвідувати платформу. Прикладом такої платформи може бути аукціон автомобілів, або навіть електроніки. Так, в середньому окремо взятий користувач може звертатися до послуг аукціону раз в декілька років.

2.4 Кластерний аналіз

Завдання кластерного аналізу полягає у виділенні на множині елементів вибірки підмножин, таких що кожен елемент підмножини максимально подібний до інших елементів даної підмножини та максимально відрізняється від елементів інших підмножин.

Завдання кластеризації відноситься до класу задач навчання без вчителя (Unsupervised learning).

При вирішенні задачі кластеризації в якості вхідних даних до алгоритму може подаватись:

— Матриця відстаней між елементами, де кожен елемент описується переліком відстаней до всіх інших елементів вибірки.

— Набір ознак (характеристик) об’єктів. Признаки можуть бути як числовими так і не числовими.

Матриця відстаней може бути виведена з описів ознак об’єктів, з використанням функцій відстані. Відповідно до цього, постановка задачі кластеризації за матрицею відстаней є більш загальною, але викристання ознак об’єктів дозволяє використовувати більш різноманітні та ефективні методи кластеризації.

2.4.1 One-hot кодування нечислових даних

При розгляді задач машинного навчання однією з важливих складових є обробка ознак об’єктів. Набори даних часто містять нечислові категорійні характеристики, а оскільки алгоритми машинного навчання здебільшого

здатні працювати лише з числовими даними важливим є попередня обробка, або кодування характеристик об’єкта.

One-hot кодування є одним з поширених методів приведення текстових або категорійних ознак до числового вигляду. Принцип роботи даного методу полягає в розбитті однієї характеристики елемента, що містить категорійні дані, на декілька. Кількість утворених характеристик дорівнює кількості можливих унікальних значень початкової характеристики. Значення нових характеристик може встановлюватись в 0 або 1, де 1 означає наявність в об’єкта конкретної ознаки. Використання даного методу є корисним при розв’язанні задач кластеризації даних, з нечисловими характеристиками, адже значення характеристик, утворених в результаті кодування є рівновіддаленими одне від одного та не впливають на точність обчислення відстані між елементами вибірки.

2.4.2 Зменшення розмірності даних з використанням методу PCA

Задача кластеризації багатовимірних даних має ряд проблем, спричинених розмірністю вихідних даних.

Багатовимірні дані є складними для сприйняття, аналізу та візуалізації. Також у зв’язку з експоненціальним ростом числа можливих значень при збільшенні розмірності повний перерахунок усіх підпросторів стає нерозв’язною задачею. Ця проблема також відома як «прокляття розмірності» [21].

Обчислення відстані між елементами дає менш точні результати для багатовимірних даних, оскільки відстань між будь-якими двома точками вибірки може збігатися. До того ж, при аналізі таких даних та подальшої кластеризації на основі значень характеристик елементів не всі ознаки елемента несуть в собі якийсь зміст, тобто їх значення не впливає на результат кластеризації, та як наслідок сприяє появі шумів у вибірці.

Одним з варіантів вирішення проблеми кластеризації багатовимірних даних є зменшення їх розмірності шляхом визначення головних компонент елемента, та відкидання незначних ознак. Метод головних компонент (PCA, Principal Component Analysis) – є одним з основних методів зменшення розмірності даних, з втратою найменшої кількості інформації. Обчислення головних компонент може бути зведене до обчислення сингулярного розкладу матриці даних або до обчислення власних векторів і власних чисел коваріаційної матриці початкових даних.

Математичний зміст методу головних компонент полягає у спектральному розкладі коваріаційної матриці C, тобто представленні простору даних у вигляді суми взаємо ортогональних підпросторів, а матриці C у вигляді лінійної комбінації ортогональних проекцій на отримані підпростори з відповідними коефіцієнтами. Коваріаційною матрицею є матриця, кожен (i, j)-елемент якої є кореляцією ознак (X_i, X_j) вихідного вектора X.

2.4.3 K-Means

K-means є одним з найпопулярніших методів кластеризації. В якості вхідних параметрів даний метод приймає кількість кластерів, а також опціонально, початкові координати центрів кластерів.

Принцип його роботи можна описати наступним чином. Спочатку вибирається кількість бажаних кластерів і призначаються початкові значення центрів кластерів. Потім кожному об'єктові з набору даних призначається кластер з найближчим центром, після чого обчислюються нові центри для кожного кластера. Кроки повторюються ітераційно до тих пір, поки не буде задоволено критерій зупинки або конвергенції [17].

До переваг даного методу можна віднести наступне:

— Простота використання методу.

— Низька складність обчислень, що приблизно дорівнює O(t*k*n*d), де k – кількість кластерів, t – кількість ітерацій, n – кількість об’єктів розмірності d у вибірці.

Недоліки:

— Класичні реалізації K-Means погано працюють з кластерами довільних форм, відмінних від сферичних.

— Вимагає самостійного визначення кількості кластерів.

2.4.4 DBSCAN

Ключова ідея алгоритму DBSCAN полягає в тому, що для кожного елемент даних кластера, сусідство даного

Радіус повинен містити принаймні мінімальну кількість даних

тобто щільність в околицях повинна перевищувати

певний попередньо визначений поріг [19]. Цей алгоритм приймає два вхідних параметри: мінімальна кількість елементів m у будь-якому

кластері та порогове значення відстані ε, що визначає поле схожих елементів, окремо взятого елементу. Кількість кластерів,

k, визначається самим алгоритмом. Основним критерієм об’єднання елементів у кластери при використанні даного алгоритму є щільність розміщення елементів. На початку роботи алгоритм знаходить кореневі елементи, тобто такі, що мають m елементів на відстані ε від себе. Усі сусідні елементи записуються до одного кластера. Даний крок повторюється для кожного сусіднього елементу.

Відстань розраховується за наперед визначеною функцією. Поширеним є використання Евклідової відстані. Евклідова відстань d між двома елементами a та b розмірності n обчислюється за формулою:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}. \quad (2.5)$$

Особливістю даного алгоритму є здатність виявляти аномальні відхилення у вибірці даних.

До недоліків даного алгоритму можна віднести складність реалізації на базі паралельних обчислень та низьку ефективність роботи в умовах малої вибірки даних, або при значній розрідженості даних у вибірці.

2.5 Баєсівскі мережі

Баєсівська мережа є графовою ймовірнісною моделлю, що представляє з себе множину подій та їх ймовірнісних залежностей. Такі моделі зазвичай використовуються в теорії ймовірностей, статистиці та в машинному навчанні.

Баєсівська мережа є направленим ациклічним графом, кожній вершині якого ставиться у відповідність випадкова подія, а ребра графа кодують відношення відносної незалежності між цими подіями.

2.6 Висновки до розділу 2

В межах даного розділу було розглянуто основні теоретичні відомості про предметну область та відомості про об'єкт проектування, а саме поняття про інтернет-аукціон та рекомендаційні системи. Виходячи з отриманої інформації можна зробити висновок про важливість механізмів надання рекомендацій для бізнесу. Рекомендаційні системи допомагають як власне комерційним платформам, на кшталт інтернет-магазинів та аукціонів, так і їх користувачам. Надання релевантних рекомендацій полегшує пошук потрібних товарів, послуг, або контенту для споживача тим самим збільшуючи його лояльність до платформи в цілому.

3 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

3.1 Вступ

В даному розділі буде розглянуто існуючі рішення в області рекомендаційних систем на прикладі реалізацій, пропонованих такими компаніями як Amazon та eBay. Причиною вибору рішень саме цих компаній пояснюється їх популярністю серед користувачів. Amazon є найбільшим інтернет-магазином на момент написання роботи, а впроваджені ними механізми створення рекомендацій було розроблено з урахуванням переваг та недоліків класичних методів. eBay є найбільшим інтернет-аукціоном широкого призначення, тому ознайомлення з методами надання рекомендацій, впровадженими даною компанією є корисним для розуміння специфіки аукціону в контексті рекомендаційної системи.

3.2 Рекомендаційна система Amazon

3.2.1 Загальні відомості

Amazon є найбільшим інтернаціональним інтернет магазином, що має понад 197 мільйонів активних користувачів на день клієнтів [11] і більш ніж 562 мільйони позицій у каталозі [12]. Враховуючи обсяги даних, для забезпечення генерації рекомендацій за прийнятний час, алгоритм надання рекомендацій повинен виконувати найдорожчі обчислення оф лайн. Як було вже розглянуто в розділах 2.2, 2.3 та 2.4, класичні методи створення рекомендацій мають ряд недоліків, в тому числі складність їх масштабування на великих об’ємах даних.

Для вирішення цієї проблеми компанією Amazon було розроблено модифікацію класичної колаборaційної фільтрації – Item-to-Item Collaborative Filtering [20].

Якість рекомендацій в значній мірі залежить від того, що ми розуміємо під "спорідненістю". Коли ми спостерігаємо, що клієнти купили як P1, так і P2, ми можемо задатися питанням, скільки покупців товару P1 випадково придбали би P2, якщо ці два товари не були пов'язані між собою. Система рекомендацій - це, в кінцевому рахунку, застосування статистики. Людська поведінка є значною мірою випадковою, тому завдання рекомендаційних систем полягає в тому, щоб виявити корисні моделі серед випадковостей.

Можна відмітити таку особливість, що для практично будь-яких двох позицій P1 та P2 користувач U1, що цікавився предметом P1, з більшою імовірністю зацікавиться також і P2 порівняно з усіма іншими користувачами. Якщо ми відберемо усіх користувачів, що придбали P1, користувач U1 гарантовано потрапить до вибірки. Аналогічним чином, користувач, який здійснив 1000 покупок, з приблизно в 50 разів більшою імовірністю потрапить до вибірки, ніж користувач U2 з 20 покупками. Урахування лише однієї випадкової покупки не дає універсальної ймовірності вибору користувачами того чи іншого товару. Отже, ми маємо зразок з певним упередженням. Для будь-якого товару X, покупці, що вже придбали X, ймовірно, зробили покупок більше, ніж середньостатистичний користувач.

На відміну від традиційних реалізацій методу колаборацйної фільтрації, метод Item-to-Item виконує більшість обчислень оф лайн та дозволяє виконувати онлайн обчислення з мінімальними витратами ресурсів, незалежно від кількості користувачів та елементів у каталозі продукції. Даний алгоритм створює рекомендації в реальному часі, масштабує масивні набори даних та генерує високоякісні рекомендації. Ілюстрація системи, що реалізує даний підхід наведено на рисунку 3.1.

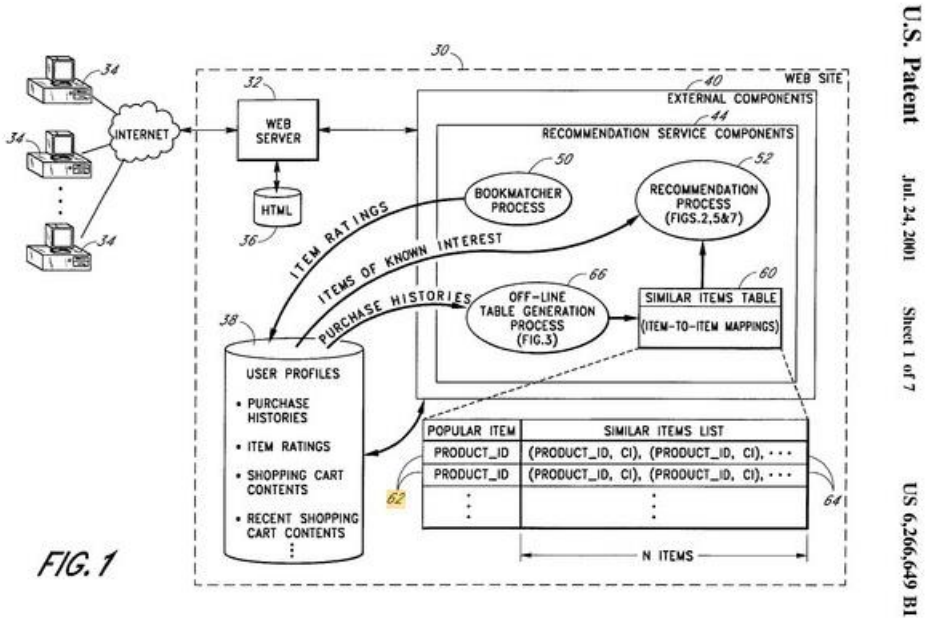


Рисунок 3.1 – Структура рекомендаційної системи на базі Item-to-Item Collaborative Filtering[1]

Найбільш витратною операцією даного алгоритму є створення таблиці споріднених елементів. Для забезпечення швидкості генерації рекомендацій в момент надходження запиту, дана операція виконується у фоновому режимі. Частиною алгоритму, що виконується в момент надходження запиту є пошук подібних до покупок оцінених товарів користувача, використовуючи дані таблиці. Складність даної операції залежить лише від кількості товарів, що були користувачем придбані або оцінені. Таким чином, алгоритм є швидким навіть для надзвичайно великої кількості даних. Також даний підхід ефективно працює з обмеженими даними користувача, надаючи релевантні рекомендації на основі лише двох-трьох пунктів, на відміну від традиційних методів колаборацийної фільтрації, алгоритм також.

[1] https://patents.google.com/patent/US6266649B1/en

3.2.2 Принцип роботи Item-to-Item Collaborative Filtering

Даний підхід полягає в пошуку пар товарів до вже оцінених або придбаних користувачем. З отриманих елементів в подальшому формуються рекомендаційні списки. Для визначення найбільш схожої пари до елементу, алгоритм будує таблицю схожих елементів [13], використовуючи інформацію про те, які товари користувачі прагнуть купувати разом. Для обчислення схожості між окремим товаром та всіма пов’язаними з ним використовується ітеративний алгоритм, схема якого наведена на рисунку. 3.2.

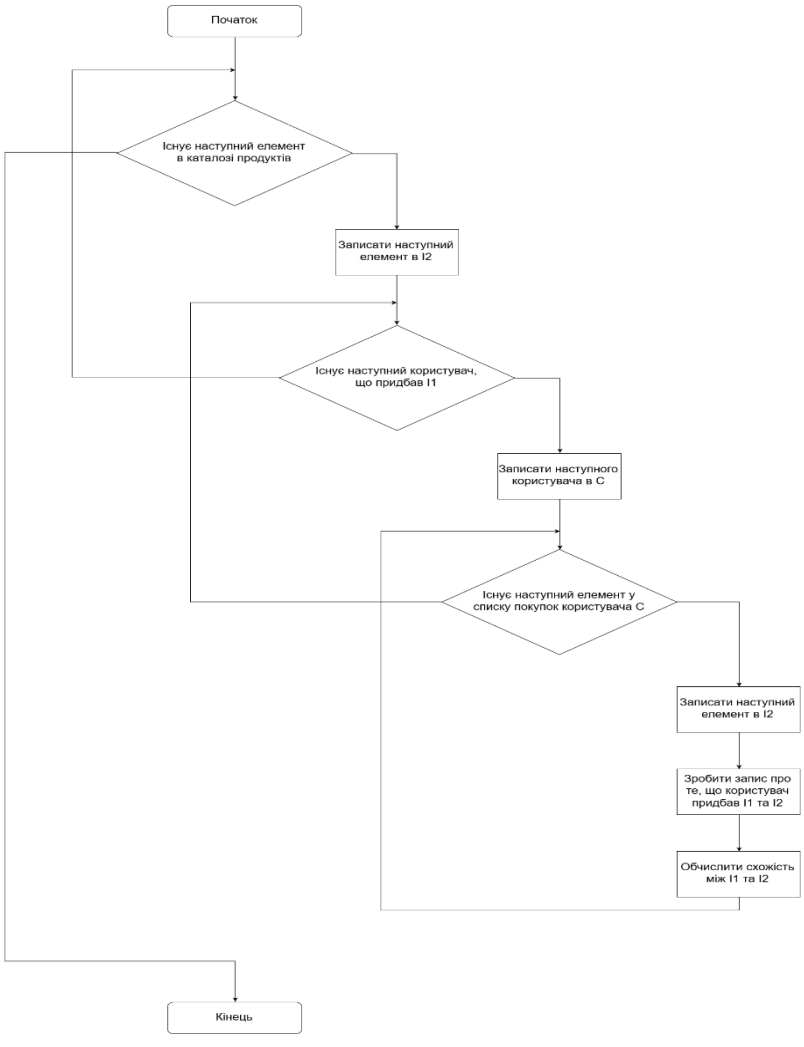


Рисунок 3.2 – Алгоритм пошуку схожих товарів

Дані розрахунки потребують надзвичайно багато часу ($O(N^2M)$) та виконуються оффлаМаючи таблицю схожості товарів, алгоритм знаходить товари, схожі до кожного з куплених або оцінених користувачем, збирає їх та пізніше рекомендує найбільш популярні, або корельовані товари. Цей етап обчислень не є складним та виконується онлайн. Складність обчислення залежить лише від кількості куплених та оцінених конкретним користувачем товарів.

3.2.3 Матрична факторизація

Для вирішення деяких обмежень фільтрації на основі вмісту, колаборaційна фільтрація використовує для надання рекомендацій одночасно подібність між користувачами та між товарами. Це дає змогу рекомендувати нові товари, що не потрапили до явної зони інтересів. Тобто, моделі колаборaційної фільтрації можуть надавати рекомендації користувачеві А, виходячи з інтересів подібного до нього користувача Б. Крім того, зв’язування можна товарів та користувачів може виконуватись автоматично.

Матрична факторизація - це доволі проста модель зв'язування товарів та користувачів. Виходячи з матриці зворотного зв'язку $A \in R^{m*n}$A, де m – це кількість користувачів, а n – кількість товарів, модель формує:

Матрицю зв'язувань користувача $U \in R^{m*d}$, де кожен i рядок відповідає зв’язкам користувача i.

Матрицю зв'язувань товару $V \in R^{n*d}$, де кожен j рядок відповідає зв’язкам товару j.

Зв'язування виконується таким чином, що добуток UV^T є хорошою апроксимацією матриці A. Приклад матричної факторизації зображено на рисунку 4.3, де P1, P2, P3, P4 – товари, U1, U2, U3. U4 – користувачі.

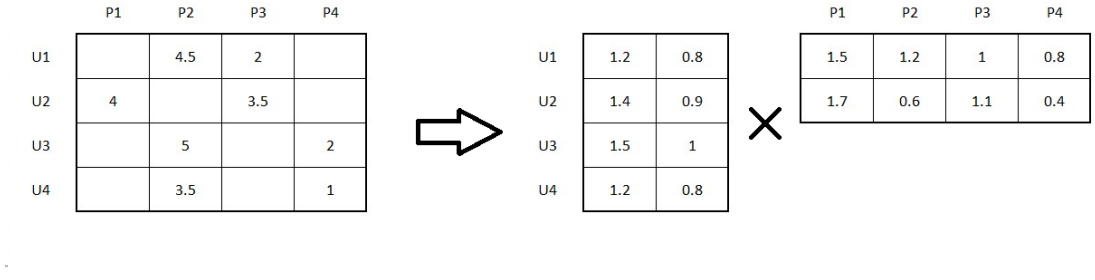


Рисунок 3.3 – Приклад матричної факторизації

Перевагою матричної факторизації є те, що даний метод дає у результаті компактніше представлення ніж повна матриця інтересів. Повна матриця зазвичай містить $m * n$ записів, в той час, коли матриці, отримані у результаті факторизації містять всього $(n + m) * d$ записів, де d – розмірність зв’язування, яка на практиці є значно меншою ніж m та n. Як наслідок, матрична факторизація дозволяє знаходити неявні залежності в даних, оперуючи з меншими розмірностями.

3.3 Рекомендаційна система eBay

3.3.1 Загальні відомості

Заснована в 1995 році, eBay є багатонаціональною e-commerce компанією, яка надає послуги онлайн-продажів у форматі C2C. Основною діяльністю eBay є аукціонна платформа та торговий сайт, через який люди та підприємства можуть купувати та продавати широкий спектр продуктів та послуг у всьому світі [14]. Кількість активних користувачів даної платформи на момент першого кварталу 2019 року сягала 180 мільйонів [15]. Також до проблеми великої кількості користувачів додаються особливості інтернет

аукціонів (дивитися розділ 2.1), що значною мірою ускладнює можливість надання рекомендацій.

Основним завданням рекомендаційної системи платформи eBay є рекомендація товарів, що доповнювали б вже придбані користувачем товари. Для адаптації методу колаборацийної фільтрації до особливостей аукціону розробниками рекомендаційної системи eBay було використано агрегацію окремих товарів в категорії, формуючи таким чином матрицю користувач-категорія. Після формування матриці для кожної категорії знаходяться пов’язані з нею шляхом обчислення схожості за косинусом, з урахуванням порогу схожості. З отриманих споріднених категорій формується поле для подальшого пошуку елементів для рекомендації.

Генерація елементів-кандидатів для рекомендації відбувається з використанням інформації про вже придбаний користувачем товар. Системою використовується декілька методів генерації кожен з яких використовує різну інформацію про поведінку користувача та має свій пріоритет. Також, варто виділити, що кожен метод використовується в залежності від наявності в історії дій користувача потрібної інформації та результатів роботи попередньо застосованих методів.

Ще одним методом, що використовується для узагальнення подібних товарів, є метод агрегації товару у продукт. Продукт в даному контексті прирівнюється до ISBN коду. Дане узагальнення дозволяє використовувати методику відбору рекомендацій, подібну до вищезгаданої. Після знаходження пов’язаних продуктів, з них відбираються елементи з найбільшою кількістю переглядів.

3.3.2 Граф інтересів користувачів

Отримані таким чином узагальнення використовуються для побудови персонального графу інтересів користувача. Розглянемо наступний приклад такого графу.

На рисунку 3.4 зображено користувачі U1, U2, U3 та товари P1, P2, P3. Користувачеві U1 подобаються P1 та P2. Користувачеві U2 не подобаються P1

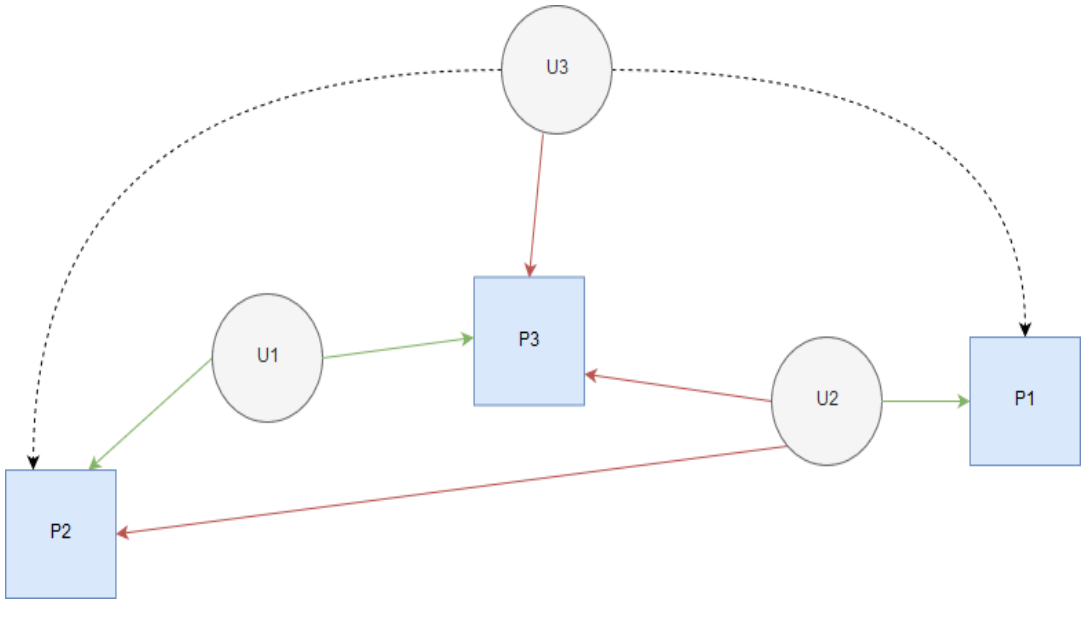


Рисунок 3.4

та P2, але подобається P3. Користувачеві U3 також не подобається P3. Виходячи з припущення кореляції даних ознак інтересів, ми можемо сказати, що користувач U3, як і користувач U2 надав би перевагу товарові P1, аніж P2, адже вони обоє не люблять P3. Подібним чином ми можемо прогнозувати зацікавленість користувача в певному предметі, базуючись на вподобаннях його та інших користувачів.

Тепер припустимо що в нас є більше користувачів, а їх ставлення до предметів зобразимо у вигляді вектору (a, b), де a – зацікавленість користувача в P3, b – зацікавленість у P2. Зацікавленість вимірюватимемо від -2, що означає не цікавить та до 2, що відповідно сигналізує про наявність інтересу. Розташуємо дані вектори на площині координат, як показано на рисунку 3.5.

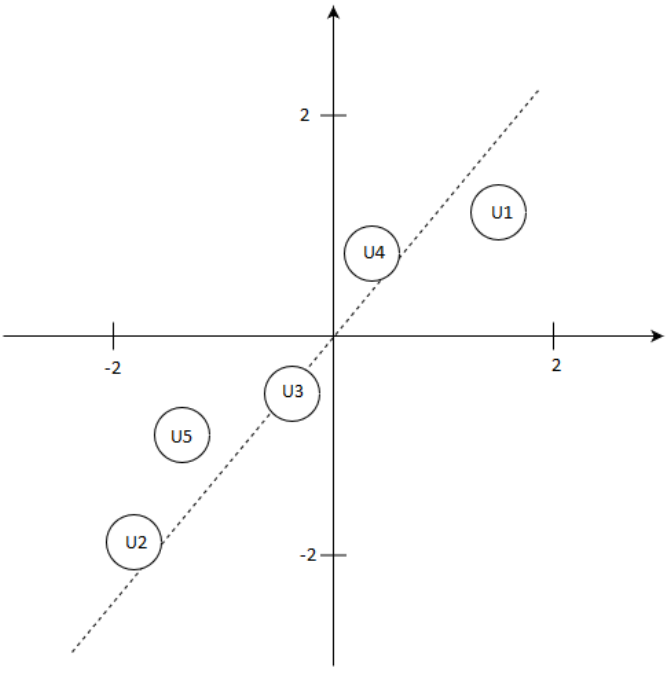


Рисунок 3.5

Можна помітити певну закономірність у розміщенні користувацьких векторів на даній площині. Імовірно існує певний прихований фактор, який може пояснити зацікавленість користувача. Ми робимо припущення, що зацікавленість користувача тим чи іншим предметом в межах даного прикладу залежить від того, чи має користувач певну, не явну ознаку чи ні. Знаходження такого прихованого фактору та його обчислення допомагає зменшити розмірність оброблюваних даних, що відповідно зменшує витрати у часі на виконання розрахунків.

eBay використовує значну кількість прихованих факторів, формуючи з них вектор смаків:

$$\vec{T} = (l_1, \dots, l_n) \tag{3.1}$$

де l_n – значення прихованого фактору, n – кількість прихованих факторів.

Даний вектор розраховується для кожного профілю товару та користувача, та використовується для обчислення схожості товарів, та

відповідності товару інтересам користувачів. Оновлення значення даного вектору відбувається при кожній покупці або оцінці користувачем товару.

Для прикладу розрахунку вектора прихованих факторів позначимо ваги зв'язків та значення прихованих факторів користувачів на рисунку 3.6.

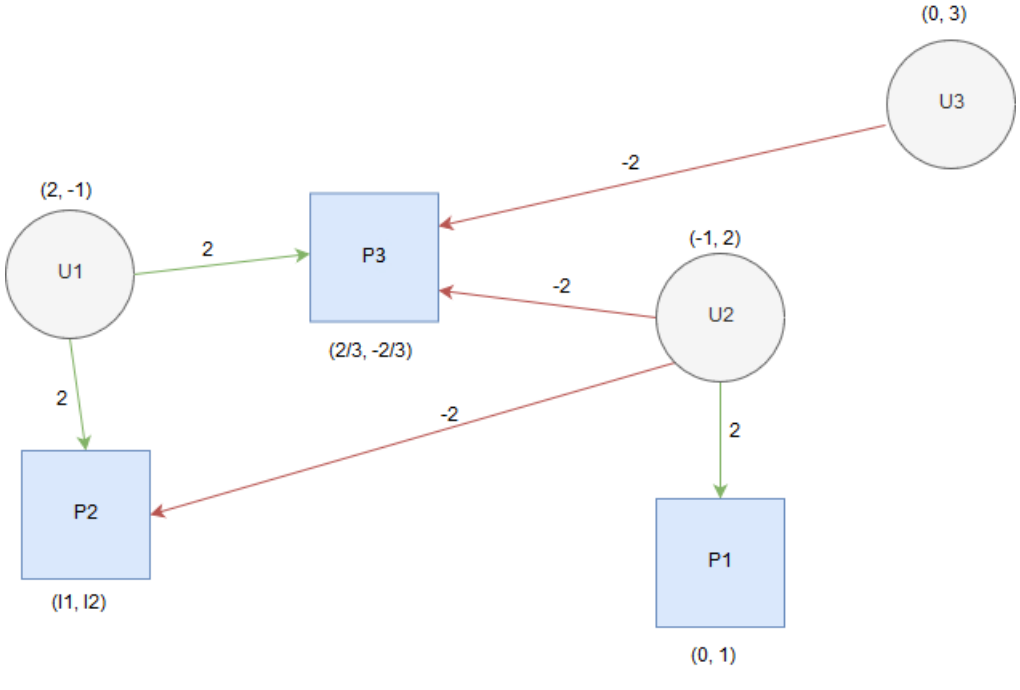


Рисунок 3.6

Значення l1 та l2 для товару P2 обчислюється наступним чином:

$$\begin{cases} -1l_1 + 2l_2 = -2 \\ 2l_1 - 1l_2 = 2 \end{cases} \rightarrow T_{P2} = (\frac{2}{3}, -\frac{2}{3}), \quad (3.2)$$

де T_{P2} – вектор прихованих факторів товару P2.

Якщо користувач U3 виявить інтерес до товару P2 придбавши його, або оцінивши, значення T_{P2} буде перераховано, з урахуванням ваги ребра між P2 та U3, а також значення відповідного вектора прихованих факторів користувача U3. У разі якщо користувач U3 придбає даний товар $T_{P2} = (l_1, l_2)$ та $T_{U3} = (l_3, l_4)$ можна буде обчислити, розв'язавши систему рівнянь:

$$\begin{cases} -1l_1 + 2l_2 = -2 \\ 2l_1 - 1l_2 = 2 \\ l_1 l_3 - l_2 l_4 = 2 \\ \frac{2}{3} l_3 - \frac{2}{3} l_4 = -2 \end{cases} \quad (3.3)$$

3.3.3 Прогнозування імовірності зацікавлення користувача рекомендацією

Для прогнозування того, чи перейде користувач на сторінку з рекомендованим товаром чи ні, або іншими словами для прогнозування інтересу користувача до пропонованого товару використовується бінарний класифікатор. В якості міток для навчання класифікатора було використано дані про попередні кліки користувача. Така стратегія показала себе не найкращим чином з тих причин, що патерни кліків користувачів різняться в залежності від того, з якою причиною користувач відвідав сайт. Відповідно, при використанні таких даних, тренувальна вибірка містить велику кількість шумів.

3.3.4 Використання характеристик товарів

Модель працює з такими двома типами характеристик елементів:

— Порівняльні характеристики.

— Якісні характеристики.

Порівняльні характеристики отримуються в результаті порівняння властивостей цільового елемента з кандидатом до рекомендації. До таких властивостей відносяться здебільшого ціна, назва і тд. Якісні властивості товару вводяться, для забезпечення користувача вищою якістю товару, вказаного в рекомендації. До них відноситься популярність, рейтинг продавця і тд.

Важливою порівняльною характеристикою є ціна. Ціни на ідентичні предмети можуть варіюватись, враховуючи відсутність централізованого механізму ціноутворення в умовах аукціону. Замість ,розрахунку різниці між ціною цільового елементу та рекомендованого системою розраховується співвідношення цих значень. В подальшому рейтинг ціни елементу розраховується з використанням нормалізованого розподілу Коші.

Використання порівняльних та якісних характеристик дозволяє відібрати найбільш підходящі товари для користувача на етапі впорядкування (ранжування) вибірки кандидатів на рекомендацію.

3.4 Висновки до розділу 3

У результаті огляду існуючих рішень можна виділити наступні особливості проектування рекомендаційних систем для комерційних платформ. Поширеним методом фільтрації даних є метод колаборaційної фільтрації. Варто відзначити, що класичні реалізації даного методу не використовуються, враховуючи кількість користувачів цільової платформи та кількість доступних товарів. Натомість використовуються модифікації даного методу для усунення частини недоліків класичних реалізацій. Одним з таких недоліків є неможливість ефективно працювати з даними великого обсягу. Для вирішення даної проблеми метод колаборaційної фільтрації модифікується таким чином, щоб було можливим виконувати найважчі обчислення у фоновому режимі. Ще одним способом оптимізації часу виконання обчислень є побудова та подальше використання графу інтересів користувачів, з метою виявлення прихованих факторів та залежностей між вподобаннями користувачів платформи.

Також, було розглянуто способи реалізації рекомендаційної системи в умовах інтернет аукціону, з урахуванням такої особливості даної предметної області, як висока текучість позицій каталогу. Рішення, запропоноване

розробниками рекомендаційної системи інтернет аукціону eBay полягає у виділенні своєрідного профіля товару, шляхом агрегації його до більш узагальнених сутностей. eBay використовують агрегацію товарів у категорії та продукти, та в подальшому оперують взаємозв'язками та відношеннями між ними для надання користувачеві рекомендацій, щодо товарів які купують разом.

Рекомендаційні системи Amazon та eBay при створенні рекомендацій опираються здебільшого саме на суміжні товари, тобто ті, які часто купують разом. Також відзначимо, що рішення цих компаній базуються на методах колаборацйної фільтрації та є модифікаціями класичних їх реалізацій. Тобто, в процесі створення рекомендацій ці системи опираються здебільшого на сформований профіль користувача. Як наслідок, система не може надавати рекомендацій користувачеві без історії покупок. Для вирішення даної проблеми Amazon в своїй системі використовують модель рекомендації доповнюючих товарів до тих, що були додані користувачем до кошика. eBay для вирішення такої проблеми використовує механізм рекомендації на базі переглянутих товарів.

4 ПРОЕКТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

4.1 Структура рекомендаційної системи

Розроблювана рекомендаційна система складається з двох підсистем:

— Підсистема генерації кандидатів

— Підсистема впорядкування кандидатів

Завданням підсистеми генерації кандидатів є побудова профілю користувача, впорядкування та групування існуючих лотів аукціону за характеристиками з подальшим їх співставленням з профілем користувача для відбору найбільш релевантних елементів. Підсистема впорядкування приймає результат роботи підсистеми генерації кандидатів та впорядковує їх за визначеними критеріями, після чого з отриманого набору даних відбираються перші n елементів та повертаються користувачеві в якості рекомендованих лотів.

Особливістю інтернет аукціону є те, що каталог товарів не є постійним та поповнюється користувачами. Кожен лот є доступним лише на короткий час. Також варто відмітити, що кожен лот існує в єдиному екземплярі. Для рекомендаційної системи це означає неможливість безпосереднього використання класичних методів рекомендації. В результаті фільтрації до рекомендованих, можуть потрапити вже не існуючі на даний момент лоти. Вирішенням такої проблеми в рамках даного проекту є розбиття каталогу товарів на групи (кластери) схожих між собою елементів.

В основі розроблюваної рекомендаційної системи лежить граф суміжних переглядів. В вузли графу відповідають за лоти аукціону та їх кластери. Виділення лотів у кластери розглянуто у розділі 4.4. Ребра графу є направленими та у випадку з вузлами, що відповідають за кластери (сегменти) лотів, мають вагу. Значення ваги обчислюється як ймовірність придбання лоту

з сегменту А після перегляду лоту сегменту В. В даному прикладі напрям зв’язку між вузлами графу буде спрямовано від вузла А до В. Додавання нових вузлів та перерозподіл значень ваги ребер здійснюється у фоновому режимі. Більш детально граф суміжних переглядів буде розглянуто у розділах 4.7.1 та 4.7.3.

4.2 Структура підсистеми генерації кандидатів

Виходячи з умов та вимог, описаних у попередніх розділах, розроблено процес створення рекомендацій підсистемою генерації кандидатів. Огляд цього процесу зображено на рисунку 4.1.

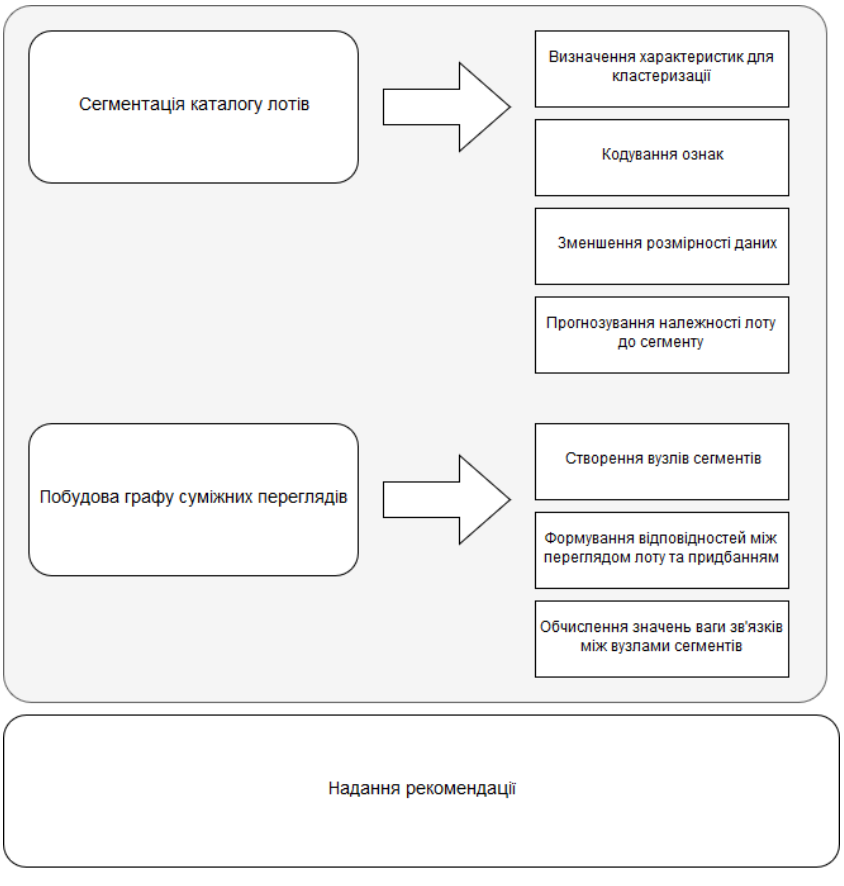


Рисунок 4.1 – Схема процесу формування рекомендацій

Процес поділяється на вісім основних кроків. Перші сім з них є фоновими процесами та виконуються періодично, в мірі надходження повідомлень про відповідні події. Розглянемо кожен крок детальніше.

Етап сегментації каталогу лотів аукціону є першим етапом в створенні рекомендацій. Для даного етапу можна виділити два сценарії: ініціалізація сегментів та обробка нових лотів.

Сценарій ініціалізації сегментів виконується лише за умови зміни структури лотів та потребує початкового набору даних, для виділення серед них кластерів за характеристиками елементів. В результаті його виконання

Рисунок 4.2 – Блок-схема виконання сценарію ініціалізації сегментів

отримується набір сегментів та модель кластеризації, що дозволяє визначити належність лоту до того чи іншого сегменту, виходячи з його характеристик. План виконання сценарію ініціалізації сегментів приведено на рисунку 4.2

Сценарій обробки нового лоту опирається на отриману, в результаті ініціалізації сегментів модель кластеризації. Виконання даного сценарію ініціюється кожного разу, коли до системи надходить повідомлення про створення нового лоту. План виконання сценарію обробки створення нового лоту приведено на рисунку 4.3.

Рисунок 4.3 – Блок-схема виконання сценарію обробки створення нового лоту

Другим і ключовим етапом є побудова графу суміжних переглядів. Суть етапу полягає в обробці історії покупок та переглядів користувачів, в результаті якої отримується набір пар переглянутого та придбаного лотів. Далі

пари групуються за придбаним лотом. З отриманих груп за алгоритмом, описаним в розділі 4.6 обчислюються ймовірності придбання лоту за умови перегляду окремого лоту.

Виконання описаних операцій у фоновому режимі дозволяє зменшити час безпосередньо генерації рекомендаційної вибірки в момент надходження запиту користувача.

4.3 Інструменти та технології реалізації

4.3.1 Мова програмування Python

Python є високорівневою інтерпретованою мовою широкого призначення з динамічною типізацією. Серед особливостей даної мови виділяється простота та локанічність синтаксису, що полегшує вивчення та подальше написання коду з її використанням.

Найбільшого поширення мова Python зазнала у вирішенні задач автоматизації сценаріїв та аналізу даних. Python широко використовується в дослідницьких роботах завдяки наявності великого набору інструментів для виконання наукових розрахунків.

Вибір Python в якості мови програмування для реалізації зумовлено наявністю великої кількості готових рішень в області аналізу даних та машинного навчання, що виокремлює Python з поміж інших мов. Варто відзначити, що Python має одну з найбільших відкритих спільнот, що сприяє наявності великої кількості матеріалів для ознайомлення з можливостями мови. Також перевагою Python є наявність динамічної типізації, що сприяє швидшій розробці на етапі прототипування.

4.3.2 Бібліотека scikit-learn

Бібліотека scikit-learn є одним з проектів Google Summer of Code, та є однією з причин популяризації мови Python в наукових та дослідницьких

колах. Бібліотека містить реалізації ряду алгоритмів машинного навчання (Machine learning) та побудована на базі стеку SciPy (Scientific Python), що включає в себе:

- NumPy: пакет, що надає структури даних для роботи з багатовимірними масивами та матрицями, а також набір математичних функцій для операцій з ними.
- SciPy: фундаментальна бібліотека засобів для наукових розрахунків.
- Matplotlib: бібліотека для побудови графіків на базі двовимірної та тривимірної графіки.
- IPython: інтерактивна оболонка для мови програмування Python.
- Sympy: пакет засобів для роботи з символьними даними.
- Pandas: бібліотека, що реалізує різні структури даних та надає засоби для і аналізу.

Бібліотека scikit-learn складається з 35 модулів, що спеціалізуються на певного роду обчисленнях. Серед них наявні модуль з реалізаціями базових алгоритмів кластеризації, модулі оцінки моделі та якості прогнозів, модулі для попередньої обробки та трансформації наборів даних, модулі для зменшення розмірності даних, модулі реалізацій базових алгоритмів для вирішення задач класифікації та регресії

4.3.3 Бібліотека pandas

Pandas – це пакет Python, що надає швидкі, гнучкі та експресивні структури даних, які роблять роботу з «реляційними» або «маркованими» даними простими та інтуїтивними. Метою створення даного пакету було впровадження високорівневого засобу для потреб практичного аналізу даних у Python. Бібліотека pandas містить дві базові структури: Series та DataFrame.

Series – це маркована одновимірна структура даних, яку можна представити у вигляді одного рядка таблиці. Series є сумісною як зі звичайним

масивом python, що дозволяє звертатися до елементів за числовим індексом, так і з асоціативним масивом.

DataFrame є двовимірною маркованою структурою, що є подібною до звичайної таблиці. Подібність цих структур виражається в способі їх створення та роботі з їх елементами.

4.3.4 СУБД Neo4j

Neo4j - система управління базами даних на базі графів з відкритим вихідним кодом, реалізована на Java. Наразі вважається найпопулярнішою СУБД такого. Розробником є американська компанія Neo Technology, що веде розробку з 2003 року. Дана СУБД використовується такими відомими компаніями як:

— eBay
— Adobe
— IBM
— LinkedIn
— Walmart
— HP

Дані зберігаються у власному спеціалізованому форматі, що дозволяє ефективно працювати з даними, поданими у графовому вигляді.

Neo4j має підтримку ACID та відповідає JTA, JTS і XA. API для роботи з СУБД реалізовано для багатьох мов програмування, серед яких C#, Python, Clojure, Ruby, Java. Також реалізовано REST API.

СУБД використовує власну декларативну мову запитів – Cypher Query Language (CQL). Синтаксис Cypher є подібним до SQL, та дозволяє описувати зв’язки у графах та оперувати ними з використанням логічних та візуальних шаблонів з використанням синтаксису ASCII-Art.

Зв’язок з базою даних забезпечується протоколом Bolt. Bolt є легким та ефективним мережевим протоколом клієнт-серверної взаємодії, що призначений для роботи з базами даних.

4.4 Підготовка даних

Використання методів машинного навчання, а саме алгоритмів кластеризації потребує попередньої обробки вхідних даних та приведення всіх характеристик розгляданих об’єктів до числового виду.

Розглянемо набір даних з характеристиками автомобілів Car Features and MSRP, завантажений з ресурсу Kaggle. Для початку відкинемо не важливі для подальшого розгляду характеристики шляхом видалення колонок “ Popularity”, “ MSRP”. Набір даних містить близько 12000 записів. Кожен запис є набором ознак автомобіля, поданих у вигляді числових характеристик та текстового опису. Характеристики, описані у текстовому вигляді належать до скінченних наборів категорій, тобто є категорійними. Перелік характеристик з прикладами їх значень наведено на рисунку 4.4

Make	Model	Year	Engine ...	Engine ...	Engine ...	Transm...	Driven_...	Numbe...	Market ...	Vehicle ...	Vehicle ...	highwa...	city mpg
BMW	1 Series M	2011	premium unleaded (required)	335	6	MANUAL	rear wheel drive	2	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19
BMW	1 Series	2011	premium unleaded (required)	300	6	MANUAL	rear wheel drive	2	Luxury,Performance	Compact	Convertible	28	19
BMW	1 Series	2011	premium unleaded (required)	300	6	MANUAL	rear wheel drive	2	Luxury,High-Performance	Compact	Coupe	28	20
BMW	1 Series	2011	premium unleaded (required)	230	6	MANUAL	rear wheel drive	2	Luxury,Performance	Compact	Coupe	28	18
BMW	1 Series	2011	premium unleaded (required)	230	6	MANUAL	rear wheel drive	2	Luxury	Compact	Convertible	28	18
BMW	1 Series	2012	premium unleaded (required)	230	6	MANUAL	rear wheel drive	2	Luxury,Performance	Compact	Coupe	28	18
BMW	1 Series	2012	premium unleaded (required)	300	6	MANUAL	rear wheel drive	2	Luxury,Performance	Compact	Convertible	26	17

Рисунок 4.4 – Попередній перегляд набору даних на сайті Kaggle

Завданням даного етапу є приведення всіх нечислових характеристик до числового вигляду. Після приведення усіх характеристик до числового

вигляду їх значення потрібно нормалізувати, тобто рівномірно розподілити значення характеристик в межах визначеного діапазону.

Кожен об'єкт вихідного набору даних містить 8 текстових (категорійних) характеристик та 5 числових. Було проведено дослідження усіх наявних характеристик за допомогою засобів пакету pandas та відкинуто характеристику “Market Category” у зв’язку з великою кількістю порожніх значень, розподіл значень категорійних характеристик приведено на рисунку 4.5

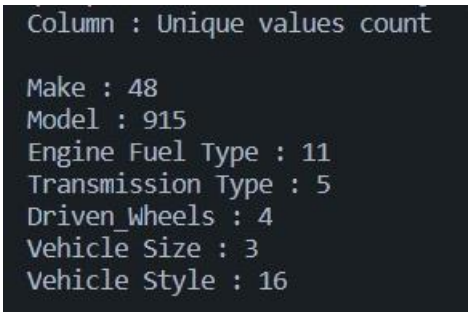


Рисунок 4.5

Далі категорійні характеристики кодуються з використанням методу кодування One-hot. Для цього використано реалізацію даного методу з пакету scikit-learn OneHotEncoder. В результаті кодування було отримано матрицю розмірності (11914, 1008). Результат можна також трактувати як 11914 векторів з 1008 вимірами в кожному. Оскільки якість автоматичної кластеризації даних таких розмірностей є невисокою, застосуємо до отриманого набору даних методи зменшення розмірності. В якості засобу зменшення розмірності взято реалізацію методу головних компонент пакету scikit-learn. Даний пакет має реалізації методів PCA та SparsePCA. Різниця між цими двома реалізаціями полягає в тому, що при використанні PCA основні компоненти зазвичай є лінійними комбінаціями всіх вхідних змінних. SparsePCA долає цей недолік, знаходячи лінійні комбінації, які містять лише кілька вхідних змінних. Таким чином SparsePCA є ефективнішим для розріджених даних.

Для забезпечення можливості візуалізації приведемо розмірність векторів до двох вимірів. Графічне представлення отриманого набору даних у вигляді точок на площині зображено на рисунку 4.6.

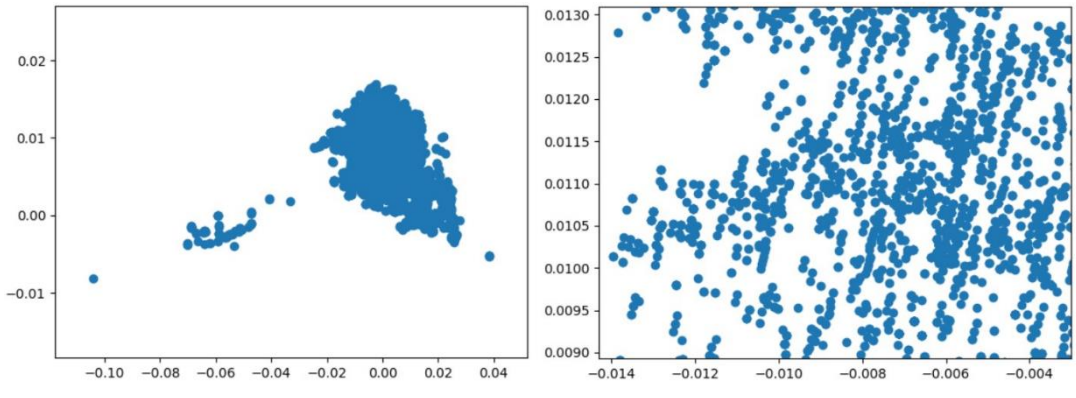


Рисунок 4.6 – Розподіл представлень лотів на площині

В результаті ми отримали компоненти моделі препроцесора лотів, що розрахована для роботи з даними, подібними для вихідного набору. Оскільки операцію підготовки даних необхідно виконувати для кожного нового лоту, було прийнято рішення про збереження отриманої моделі для подальшого використання. Таке рішення допоможе зменшити витрати часу при прогнозуванні сегменту нових лотів, так як модель не потрібно буде вираховувати з нуля для кожного лоту.

4.5 Сегментація каталогу лотів аукціону

Попередня класифікація лотів аукціону є важливим етапом роботи розроблюваної системи. Підхід з розбиттям лотів на групи є схожим до того, що використовується у при реалізації методу колаборaційної фільтрації на базі кластерної моделі. Відмінністю є предмет кластеризації. Метод колаборaційної фільтрації на базі кластерної моделі використовує розбиття користувачів на групи для надання подібних рекомендацій в межах групи. Перевагою є можливість виконувати кластеризацію у фоновому режимі. Недоліком є низька точність надання рекомендацій, а також необхідність

періодичного перерозбиття груп користувачів, оскільки поведінка користувачів може змінюватися з часом і рекомендації, що є релевантними для групи можуть не цікавити окремого користувача.

Використовуваний в рамках даного проекту підхід полягає у розбитті на групи лотів аукціону. Оскільки опис лоту є здебільшого незмінним на час його існування, достатнім буде разове виконання операції розбиття існуючих лотів на групи, з подальшим прогнозуванням належності нових лотів до тої чи іншої групи.

На рисунку 4.7 продемонстровано результат роботи реалізації даного алгоритму, взятої з python пакету scikit-learn. Для даного прикладу було обрано наступні значення вхідних параметрів:

eps=0.00024, min_samples=5.

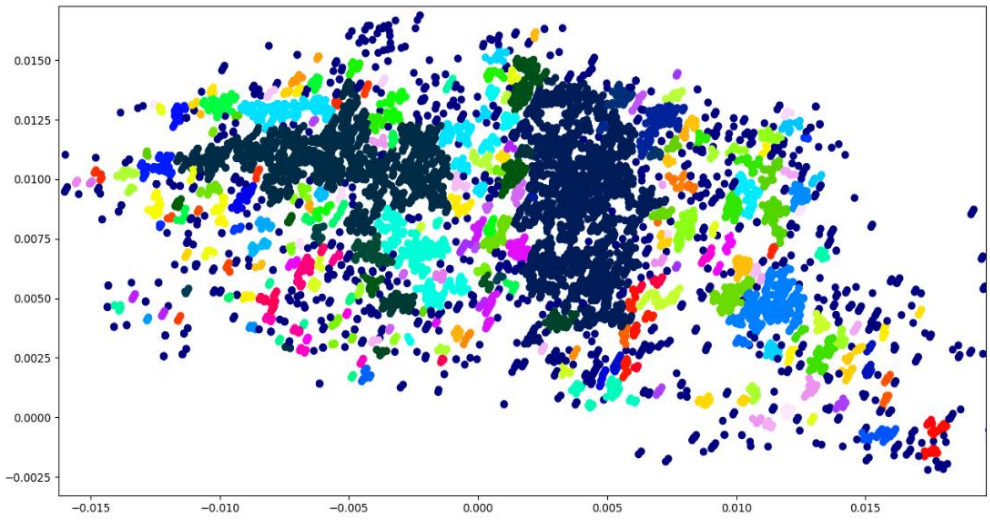


Рисунок 4.7 – Результат кластеризації методом DBSCAN

Як видно з рисунку 4.7 розбиття на кластери даної вибірки даних є нерівномірним. При загальній кількості кластерів, що наближається до 160 середньою кількістю елементів у кластері є 35. Мінімальний розмір кластера дорівнює 5 елементів, відповідно до заданого параметру, а максимальне значення розміру кластера є близьким до 1200. При зміні вхідних параметрів

ситуація не покращується. При зміні значення параметру eps більшість елементів ідентифікуються як шуми. Такі результати пов'язані з високою щільністю вихідного набору даних та як наслідок – градієнтний розподіл елементів.

Вхідним параметром алгоритму є кількість кластерів, що розраховується автоматично, виходячи з кількості наявних елементів у вибірці та мінімального розміру кластеру. Для розрахунку даного параметру було використано формулу (4.1):

$$n_{clusters} = \left\lfloor \frac{n}{c_{min}} \right\rfloor, \quad (4.1)$$

де c_{min} – мінімальна кількість елементів у кластері,

n – кількість елементів вибірки.

В результаті для мінімальної кількості в 40 елементів було отримано приблизно 300 кластерів. Приклад роботи алгоритму з використанням реалізації BatchKMeans пакету scikit-learn на вибірці даних Car features and MSRP приведено на рисунку 4.8.

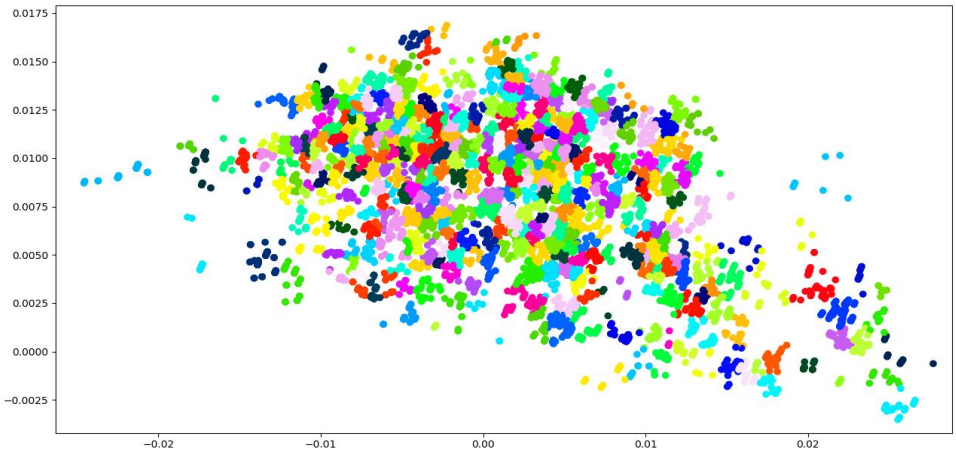


Рисунок 4.8 – Результат кластеризації методом k-means

При порівнянні цих двох методів кластеризації на даних великої розмірності та з високою щільністю вибірки можемо зробити висновок, що k-

means дає більш рівномірне розбиття на групи, тому при реалізації компоненту системи використовуватимемо саме цей метод.

4.6 Обробка історії користувачів

Надання рекомендації на основі попередніх дій користувача в загальному випадку зводиться до прогнозування вибору користувача на основі історії його взаємодії з платформою.

Взаємодію користувача з платформою зручно зобразити у вигляді послідовності визначених подій у часі. Кожну подію, в свою чергу можна описати у вигляді таких параметрів:

— Ідентифікатор користувача

— Ідентифікатор об’єкту взаємодії

— Тип події

— Час події

В даній роботі ми розглянемо такі типи подій, як перегляд та викуп лоту. Використання інформації про перегляди дозволяє відтворити хід пошуку потрібного лоту та скласти уявлення про мотивацію конкретного користувача.

В ході роботи було проаналізовано набір даних, опублікований на ресурсі Kaggle компанією Retail Rocket. Дані зібрані з реального веб-сайту електронної комерції. Набір містить необроблені дані про поведінку користувачів (рисунок 4.9), наприклад такі події, як перегляди, додавання товару до кошику та транзакції покупок.

	# timestamp the time, when event is occurred, in milliseconds since 01-01-1970 (Standard Unix timestamp	# visitorid unique identifier of the visitor	A event type of the event {"view", "addtocart", "transaction"}	# itemid unique identifier of the item	# transactionid unique identifier of the transaction (non empty only for transaction event type).
1	1433221332117	257597	view	355908	
2	1433224214164	992329	view	248676	
3	1433221999827	111016	view	318965	
4	1433221955914	483717	view	253185	
5	1433221337106	951259	view	367447	
6	1433224086234	972639	view	22556	
7	1433221923240	810725	view	443030	
8	1433223291897	794181	view	439202	
9	1433220899221	824915	view	428805	
10	1433221204592	339335	view	82389	

Рисунок 4.9 – Фрагмент набору даних

Зібрано дані про події трьох, а саме: «view», «addtocart» або «transaction». Всього налічується 2 756 101 подій, у тому числі 2 664 312 переглядів, 69 332 додавань до кошику та 22 457 транзакцій - 1 407 580 унікальних відвідувачів. Всі значення є захешованими у зв'язку з питаннями конфіденційності. Оскільки інформація про перегляди має велику кількість шумів, тобто випадкових переглядів, для їх ефективного використання необхідно провести попередню обробку існуючих даних. Також для подальшого використання згрупуємо події за покупками, звівши до таблиці наступного вигляду (рисунок 4.10).

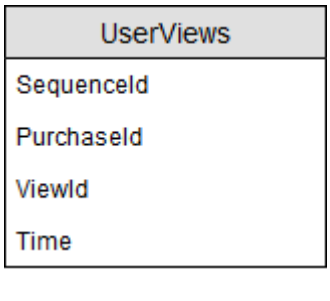


Рисунок 4.10 – Структура запису про подію в послідовності

Обробка даних виконується з використанням ітеративного алгоритму, що полягає у впорядкуванні подій за часом пошуку n останніх записів з типом події «view» для кожної події «transaction» та «addtocart». Схему алгоритму приведено на рисунку 4.11.

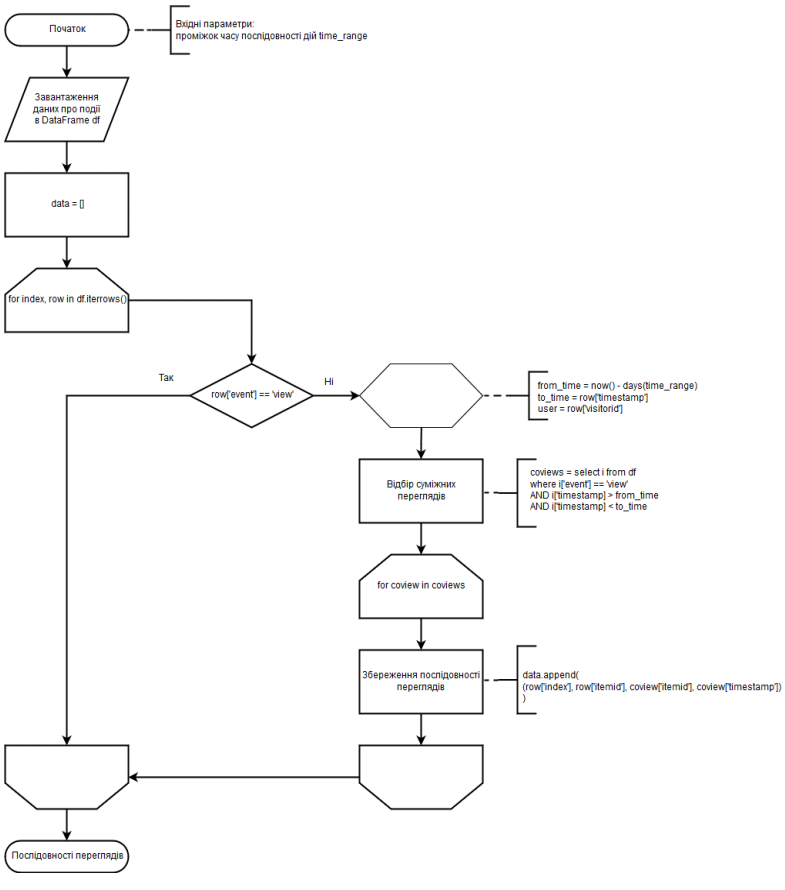


Рисунок 4.11 – Блок-схема алгориту обробки історії переглядів та покупок

В результаті роботи алгоритму отримуємо набір записів формату, приведеного на рисунку 4.10. Оскільки алгоритмом розглядались лише події, що передували придбанню товару, вибірку було профільтровано від випадкових переглядів. Також враховується час, за який ці події відбулися. До результуючої вибірки потрапили лише події, перегляди користувача, що були здійснені у проміжок часу розміром у п'ять днів перед придбанням товару. В результаті отримано 91789 груп подій, що відповідають кількості подій покупки в початковому наборі даних, кожна група в середньому містить 11 записів про перегляд.

4.7 Граф суміжних переглядів

В основі підсистеми генерації кандидатів лежить ймовірнісна графова модель представлення переглядів та покупок лотів. Граф представляє собою набір випадкових подій двох типів та їх умовні залежності. Таким чином дана модель належить до такого різновиду статистичної моделі як Баєсівська мережа. Вузли даного графу поділяються на два типи, кластер лотів (ItemsCluster) та лот (Item). Ребра графу є направленими та в свою чергу поділяються на перегляди (CoView) та ребро належності лоту до певного кластеру (BelongsTo).

Ребро типу BelongsTo відповідає за належність конкретного лоту тому чи відповідному кластеру та є направленим від вузла лоту до вузла кластера. Ребра CoView відповідають за зв'язок між кластером переглянутого лоту та придбаного. Даний зв'язок має напрям від вузла кластера переглянутого лоту до вузла, що відповідає кластеру придбаного лоту. Структуру графу зображено на прикладі рисунку 4.12.

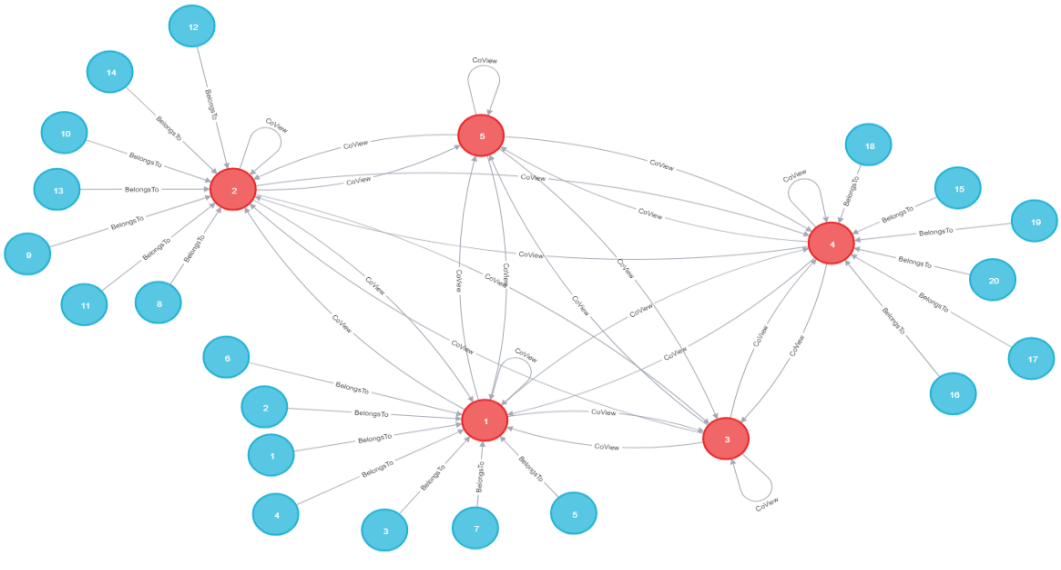


Рисунок 4.12 – Сині вузли відповідають конкретним лотам, червоні – кластерам

Кожен вузол графа має набір параметрів, що дозволяють його ідентифікувати. Так, кожен вузол, незалежно від типу має параметр id, що є

унікальним для усіх вузлів у межах типу та визначається відповідним ідентифікатором кластера або конкретного лоту. Крім цього вузли типу Item мають булевий параметр available, що визначає доступність відповідного лоту.

Для ребер типу CoView визначається їх вага. При ініціалізації графу між усіма вузлами типу ItemsCluster будуються двосторонні зв’язки на ребрах CoView, для яких значення ваги встановлюється в 0. Два вузли, з’єднані таким ребром відображають умовну імовірність $P(B|A)$, де B – подія викупу лоту з кластера, A – подія взаємодії користувача з лотом кластеру, тобто його перегляд.

4.7.1 Реєстрація викупу лоту

Визначення та перерахунок значень ваги ребер графа виконується в процесі реєстрації викупу лоту. Також потрібно врахувати те, що після викупу лоту він стає недоступним для подальшої взаємодії, а тому рекомендація такого лоту не буде релевантною. Для оброки цього сценарію до вузла лоту додається булевий параметр «available», який при створенні лоту встановлюється в значення True. Відповідно при реєстрації викупу лоту значення змінюється на False, після чого даний лот більше не враховуватиметься при генерації рекомендацій.

Перерахунок ваги ребер суміжних переглядів оновлюється наступним чином. Для прикладу розглянемо простий граф з трьома кластерами, з одним лотом в кожному та вихідними значеннями ваги ребер з рисунку 4.13.

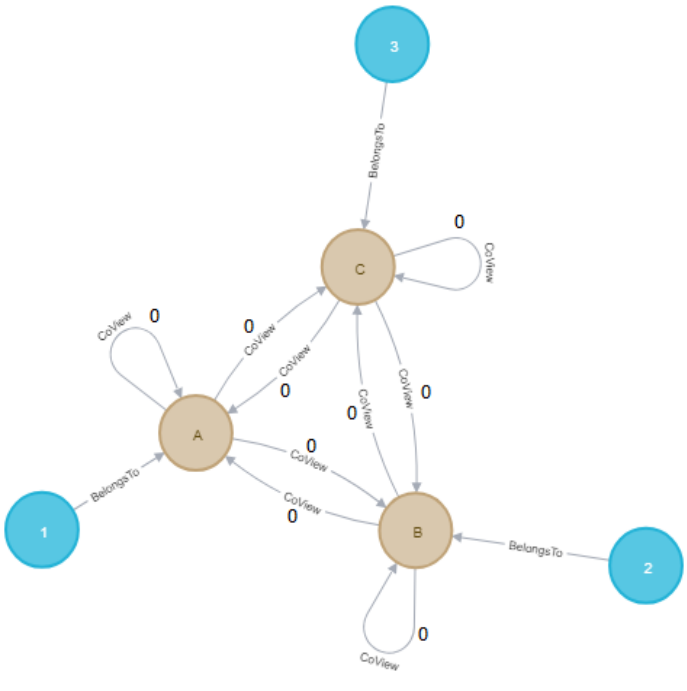


Рисунок 4.13

Маємо кластери A, B, C з лотами 1, 2, 3 відповідно. Припустимо, що в системі було викуплено лот 1. З історії користувача, що здійснив покупку нам відомо, що перед безпосередньо викупом він переглянув також лоти 3 та 1. Виходячи з цього ми можемо агрегувати лоти до кластерів та отримати наступне твердження: користувач здійснив викуп лоту кластера A, попередньо переглянувши кластери A та C. Відповідно до цієї інформації перерахуємо ваги ребер за наступною формулою:

$$W = \frac{n_{views}}{n_{purchases}}, \tag{4.2}$$

де $n_{purchases}$ – загальна кількість викуплених лотів даного кластера, n_{views} – кількість покупок лотів даного сегмента з попереднім переглядом лоту суміжного сегменту.

Узагальнимо використання даної формули для практичного використання в системі. Визначимо множину унікальних переглядів сегментів V, що передували покупці з сегмента p. Тоді, використовуючи попереднє

значення ваги ребра між сегментом p придбаного лоту та суміжного до нього x, дану формулу можна привести до вигляду:

$$W(p,x) = \frac{(W_{px0} + g(x))}{2}, \tag{4.3}$$

$$g(x) = \begin{cases} 1, x \in V \\ 0, x \notin V \end{cases}. \tag{4.4}$$

де W_{px0} – попереднє значення ваги ребра між вузлами p та x.

В результаті розрахунків отримуємо значення ваги для ребер графа, зображені на рисунку 4.14.

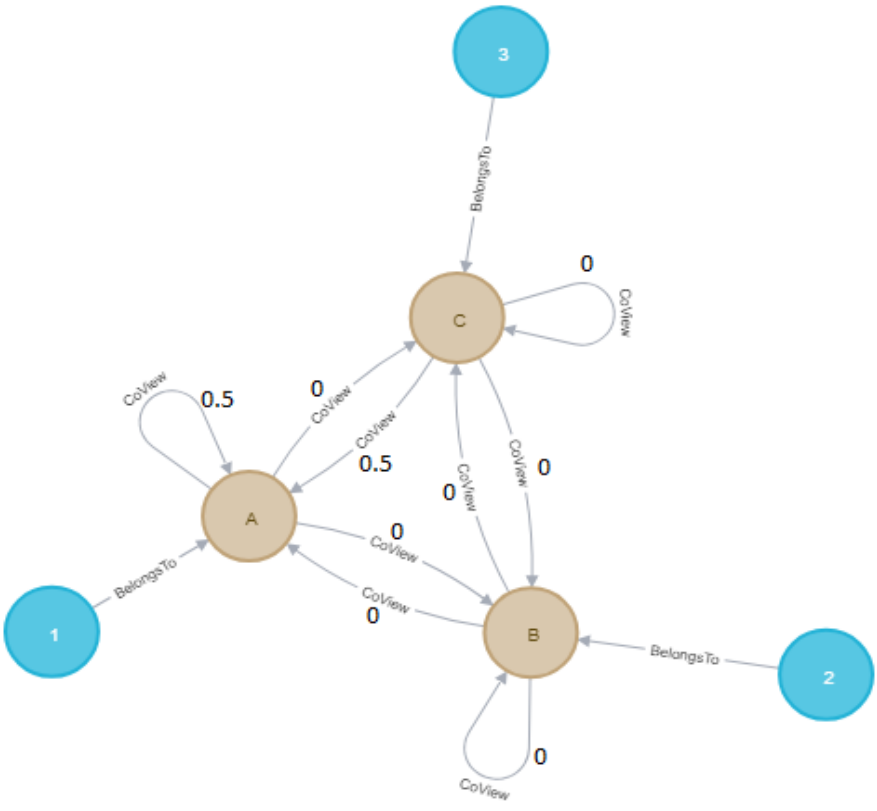


Рисунок 4.14

Використання даної операції дозволяє проводити перерозподілу ваги ребер за часом $O(n)$, де n – кількість вузлів сегментів у графі. При великій кількості сегментів операція може займати певний час, тому дані розрахунки проводяться у фоновому режимі по мірі надходження повідомлень про викуп лоту.

4.7.2 Визначення рекомендованих лотів

Для подальшого розгляду операцій над графом, а саме процесу відбору рекомендацій, збільшимо кількість сегментів та покупок, для зміни значень ваги ребер. У результаті отримуємо граф, поданий для зручності у вигляді матриці, зображеної на рисунку 4.15.

Перегляди \ Покупки	A	B	C	D	E
A	0	0.75	0	0.5	0.5
B	0.25	0.75	0	0.25	0
C	0.625	0	0	0.25	0.5
D	0.875	0	0	0	0.5
E	0	0	0	0.5	0

Рисунок 4.15 – Матричне представлення графу

Представимо вузли графа як залежні події, а саме подія придбання лоту та подія перегляду лоту, де ймовірність виникнення першої події залежить від виникнення другої. Тобто, ми можемо стверджувати, що після перегляду лоту з сегменту C користувач з імовірністю 0.625 придбає лот з сегменту A. Тепер обчислимо імовірність придбання користувачем лоту з сегменту A, якщо історія його переглядів містить лоти з сегментів C та D. Введемо позначення ваги ребра W_{NM} між деякими вузлами N та M, при напряму зв'язку від N до M. Розглянемо наступні події: A_C – покупка лоту з сегменту A після перегляду лоту з сегменту C та A_D – покупка лоту з сегменту A після перегляду лоту з сегменту D, ймовірність яких дорівнює W_{DA} та W_{CA} відповідно. Так як

приведені події є сумісними обчислимо ймовірність суми цих подій за формулою:

$$P(A_D + A_C) = P(A_D) + P(A_C) - P(A_D)P(A_C), \quad (4.5)$$

Підставимо до формули значення ваги відповідних ребер та отримаємо ймовірність придбання лоту з сегменту A за умови наявності в історії переглядів користувача лотів з сегментів D та C, що дорівнюватиме 0.953. Запишемо узагальнену формулу для обчислення ймовірності події Y придбання лоту з сегменту M, за наявності в історії переглядів лотів з сегментів $N_i, i \geq 0$, та відповідними вагами ребер W_{N_iM} :

$$P(Y) = \sum_i W_{N_iM} - \sum_{i,j} W_{N_iM}W_{N_jM} + \sum_{i,j,k} W_{N_iM}W_{N_jM}W_{N_kM} - \ldots + (-1)^{n-1} \prod W_{N_iM} \quad (4.6)$$

З метою спрощення обчислень, перейдемо до ймовірностей протилежних подій. В результаті отримаємо:

$$P(Y) = 1 - \prod (1 - W_{N_iM}) \quad (4.7)$$

Для отримання рекомендованого сегменту застосовується наступний алгоритм. Для кожного вузла графу, що відповідає переглянутому сегменту за формулою (4.7) обчислюється ймовірність придбання лоту кожного сегменту, вузол графу якого з'єднується ребром з ненульовою вагою та є спрямованим від вузла переглянутого сегменту. Після цього обирається n сегментів, ймовірність придбання для яких є найбільшою. Лоти отриманих сегментів визначаються як рекомендовані та надсилаються до підсистеми впорядкування.

4.7.3 Додавання лоту

Для створення залежностей між діями користувача та лотом до графу вносяться конкретні лоти та їх сегменти. Належність лоту до того чи іншого сегменту визначається наявністю між ними зв'язку типу BelongsTo, направленого від лоту до сегменту. Оскільки об'єктом взаємодії в процесі генерації рекомендаційної вибірки є сегменти лотів, а не самі лоти, для

додавання лоту в граф необхідно вказати до якого саме сегменту відноситься лот. Процес додавання нового лоту можна описати наступним чином (рисунок 4.16).

В ході даного процесу, окрім створення лоту в граф при потребі додається також вузол відповідного йому сегменту. Це дозволяє уникнути

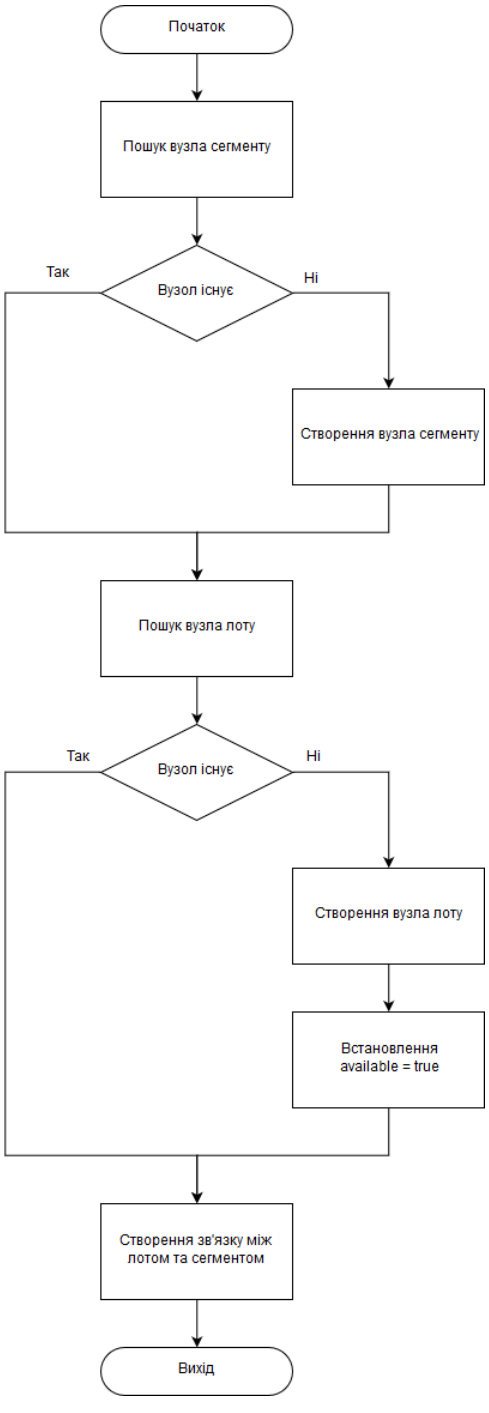


Рисунок 4.16 – Блок-схема
алгоритму додавання нового лоту

потреби в попередній ініціалізації вузлів сегментів. Також це дозволяє спростити розширення каталогу, та збільшення кількості використовуваних сегментів.

4.8 Аналіз результатів роботи моделі

Для аналізу результатів роботи було використано набір даних Retail Rocket, розглянутий раніше в розділі 4.6 Для чого дані було попередньо оброблено та відібрано записи про 10 000 покупок. Отриманий набір випадковим чином розбито на тренувальні дані та тестові по 9 066 та 934 записів відповідно. Розмір рекомендаційної вибірки дорівнює 50 записів. Загальна кількість товарів – 14829.

Результати було отримано з використанням двох варіантів розміру послідовності переглядів, обмежених у часі за п’ять днів та три дні (таблиця 4.1). В якості метрик взято загальну кількість зроблених рекомендацій та кількість точних співпадінь рекомендації та реально придбаного користувачем товару, з чого обчислено ймовірність прогнозування покупки користувача. Отримані ймовірності порівняймо з ймовірністю прогнозу при випадковому виборі 50 товарів, обчислену за формулою повної ймовірності, зведеною до вигляду:

$$P(A) = \sum_{i=1}^{n} P(A_i), \tag{4.8}$$

де n – величина вибірки, A_i – ймовірність співпадіння випадково вибраного товару з реальною покупкою.

Отримаємо ймовірність 0.0033.

Таблиця 4.1 – Результати роботи моделі

Використано історію переглядів користувача за	**Загальна кількість прогнозів**	**Кількість точних співпадінь**	**Ймовірність прогнозування покупки**
5 днів	934	870	0,931
3 дні	934	861	0,921

4.9 Висновок до розділу 4

В даному розділі було описано структуру проектованої системи та модель створення рекомендацій на основі історії переглядів користувача. Приведено алгоритми попередньої обробки даних для подальшого їх використання системою, а також розглянуто структуру моделі прогнозування покупок з використанням графу суміжних переглядів. При побудові алгоритмів підготовки даних, а також структури моделі враховувались особливості предметної області. Було отримано та проаналізовано результати роботи моделі на наборі даних, що містить історію покупок та переглядів реальних платформ електронної комерції.

ВИСНОВКИ

В результаті виконання даної роботи було розглянуто загальну будову рекомендаційних систем. Було проаналізовано класичні підходи та базові методи створення рекомендацій, для кожного з них було описано особливості, переваги та недоліки.

Методи фільтрації на базі вмісту дозволяють надавати рекомендації виходячи з опису товарів, або контенту, та знаходити подібні до вже оцінених користувачем. Головним недоліком є потреба в глибокому аналізі предметної області, що ускладнює розробку систем на основі цього методу. Також, на відміну від підходів, що базуються на методах колаборацийної фільтрації даний підхід дозволяє рекомендувати елементи лише в межах існуючих інтересів користувача.

Колаборацийна фільтрація є найбільш поширеним та популярним методом створення рекомендацій, адже дозволяє порівняно швидко розробляти рішення на його основі. Суть даного методу полягає в аналізі поведінки користувача, та пошуку прихованих факторів. Базуючись на цих даних метод рекомендує користувачеві товари, що були оцінені відповідним чином іншими користувачами зі схожими вподобаннями. Серед недоліків є проблема «холодного старту», що полягає в неможливості рекомендувати нові товари поки достатня кількість користувачів не їх не оцінить.

Було проаналізовано предметну область, а саме інтернет-аукціон, та сформовано вимоги до рекомендаційної системи аукціону. В результаті аналізу було визначено ряд вимог до рекомендаційної системи. Рекомендаційна система аукціону повинна надавати релевантні рекомендації наявних товарів в умовах динамічно-змінюваного каталогу товарів.

В результаті роботи було спроектовано систему рекомендації лотів аукціону з урахуванням особливостей комерційних платформ виду C2C та побудовано модель генерації рекомендацій на основі попередніх дій

користувача. Отримана модель використовує сегментацію за вмістом існуючих лотів, що дозволяє не прив'язуватись до конкретних товарів і як результат не залежати від часу доступності кожного окремо взятого лоту. Модель формує рекомендаційну вибірку на основі останніх переглядів користувача.

На відміну від класичних методів колаборацийної фільтрації це дозволяє надавати релевантні рекомендації новим користувачам, без попереднього формувння їх профілю. Ще однією перевагою є можливість швидко адаптуватись до змін у поведінці користувачів. Такі особливості моделі дозволяють використовувати її в умовах ринку товарів з високою вартістю, наприклад в аукціоні автомобілів. В таких умовх кожен користувач більше часу витрачає на пошук потрібного товару, а після придбання в більшості випадків покидає платформу на невизначений термін. Оцінка результатів роботи моделі показала здатність спрогнозувати покупку користувача на основі історії його переглядів з ймовірністю 0.931 порівняно з ймовірністю при випадковому розподілі в 0.0033.

Отже, результати роботи, виконаної в межах даного дипломного проекту, дозволяють побудувати систему рекомендації лотів аукціону в умовах низького рівня повторної взаємодії користувача з аукціоном.

ПЕРЕЛІК ПОСИЛАНЬ

1. Митна енциклопедія : у 3 т. / І. Г. Бережнюк (відп. ред.) та ін. — Хм. : ПП Мельник А. А, 2014. — Т. 1 : А — З. — 592 с. — ISBN 978-966-346-853-2.
2. SQA.Higher National Computing: E-Learning Materials. Business Operations on the Internet, http://www.sqa.org.uk/e-learning/ECIntro01CD/page_04.htm.
3. Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 4-6.
4. Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 10-14.
5. Machine Learning Glossary. Google Developers, Candidate generation, https://developers.google.com/machine-learning/glossary/#candidate_generation.
6. Machine Learning Glossary. Google Developers, Scoring, https://developers.google.com/machine-learning/glossary/#scoring.
7. Claudio Adrian Levinas. An Analysis of MemoryBasedCollaborative Filtering Recommender Systemswith Improvement Proposals, pp.18-19.
8. Kent Anderson. Patterns In and Across Aggregated Data — Is "Anonymous" Collaborative Filtering Really Safe?, https://scholarlykitchen.sspnet.org/2011/07/19/patterns-in-aggregated-data-is-anonymous-collaborative-filtering-safe
9. The Adaptive Web, LNCS 4321, pp. 325–341, 2007.
10. Blanda, Stephanie (May 25, 2015). "Online Recommender Systems – How Does a Website Know What I Want?". American Mathematical Society. Retrieved October *31, 2016*.

11. Statista.com. Monthly unique visitors to U.S. retail websites 2017, https://www.statista.com/statistics/271450/monthly-unique-visitors-to-us-retail-websites/.
12. Scrapehero.com. How Many Products Does Amazon Sell? – January 2018, https://www.scrapehero.com/many-products-amazon-sell-january-2018/.
13. United States patent US6266649B1. Collaborative recommendations using item-to-item similarity mappings.
14. Statista.com. eBay - Statistics & Facts, https://www.statista.com/topics/2181/ebay/.
15. Statista.com. Number of eBay's active users from 1st quarter 2010 to 1st quarter 2019 (in millions), https://www.statista.com/statistics/242235/number-of-ebays-total-active-users/.
16. Machinelearning.ru. Кластеризация, http://www.machinelearning.ru/wiki/index.php?title=Кластеризация.
17. Strategies for Big Data Clustering. 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, p. 741.
18. Machinelearning.ru. Машинное_обучение, http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение
19. Strategies for Big Data Clustering. 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, p. 742
20. Industry Report. Published by the IEEE Computer Society, JANUARY - FEBRUARY 2003, p. 78.
21. Richard Ernest Bellman; Rand Corporation (1957). Dynamic programming. Princeton University Press. ISBN 978-0-691-07951-6., Republished: Richard Ernest Bellman (2003). Dynamic Programming. Courier Dover Publications. ISBN 978-0-486-42809-3.

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
# обробка даних лоту
# fit_encoder.py

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

from scipy.sparse import issparse

def fit_encoder(data) -> ColumnTransformer:
    categorical_features = list(data.loc[:, data.dtypes ==
object].columns.values)
    categorical_transformer = Pipeline(
        steps=[
            ('imputer', SimpleImputer(strategy='constant',
fill_value='missing')),
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]
    )
    numeric_features = list(data.loc[:, data.dtypes != object].columns.values)
    numeric_transformer = Pipeline(
        steps=[
            ('imputer', SimpleImputer(strategy='median')),
            ('scaler', StandardScaler())
        ]
    )

    encoder = ColumnTransformer(
        transformers=[
            ('cat', categorical_transformer, categorical_features),
            ('num', numeric_transformer, numeric_features)
        ]
    )

    encoder.fit(data)

    return encoder

from shared.singleton import Singleton
from preprocessor import ItemPreprocessor

# classificatory.py
class ItemClassificator(metaclass=Singleton):
    def __init__(self):
```

```
        self.__clustering_model__ = None
        self.__preprocessor__ = None

    def update_model(self, clustering_model, preprocessor: ItemPreprocessor):
        self.__clustering_model__ = clustering_model
        self.__preprocessor__ = preprocessor

    def predict(self, data):
        transformed = self.__preprocessor__.transform(data)

        return self.__clustering_model__.predict(transformed)

# preprocessor.py
class ItemPreprocessor():
    def __init__(self, encoder, dimensity_reducer):
        self.__encoder__ = encoder
        self.__dimensity_reducer__ = dimensity_reducer

    def transform(self, data):
        encoded = self.__encoder__.transform(data)

        X = encoded.toarray()

        return self.__dimensity_reducer__.transform(X)

# обробка історії переглядів та покупок користувачів
# user_query_service.py

import pandas
import random
import datetime
from collections import namedtuple
from sklearn.model_selection import train_test_split

time_range = 3

CoViewEntry = namedtuple('CoViewEntry', [ 'SequenceId', 'PurchaseId', 'ViewId',
'Time' ])
CoViewSequence = namedtuple('CoViewSequence', [ 'PurchaseId', 'Views' ])

class UserQueryService():
    def fetch_queries(self, limit_purchases = 10000):
        df = pandas.read_csv('events.csv')

        df.sort_values(by=['timestamp'], ascending=[False])

        data = list()

        ver = datetime.datetime.now().timestamp()
```

```
        purchases_count = 0

        for index, row in df.iterrows():
            if row['event'] != 'view':
                purchases_count += 1

                user = row['visitorid']
                to_time = row['timestamp']

                from_time = datetime.datetime.now() -
datetime.timedelta(days=time_range)

                coviews = df.loc[(df['event'] == 'view') & (df['timestamp'] <
to_time) & (df['timestamp'] > from_time.timestamp()) & (df['visitorid'] == user)]

                coviews.drop_duplicates(subset='itemid', keep='last',
inplace=True)

                for _, coview in coviews.iterrows():
                    data.append((CoViewEntry(SequenceId=index,
PurchaseId=row['itemid'], ViewId=coview['itemid'], Time=coview['timestamp'])))

            if purchases_count == limit_purchases:
                break

        df = pandas.DataFrame(data, columns =[ 'SequenceId', 'PurchaseId',
'ViewId', 'Time' ])
        df.to_csv('data_%s.csv' % ( ver ))

        sequences = dict()

        for _, row in df.iterrows():
            seq = sequences.get(row['SequenceId'],
CoViewSequence(PurchaseId=row['PurchaseId'], Views=list()))
            seq.Views.append(row['ViewId'])
            sequences[row['SequenceId']] = seq

# опрацювання зміни структури каталогу
# dataset_changed_handler.py

import pandas

import numpy

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.decomposition import SparsePCA, PCA
```

```
from sklearn.cluster import MiniBatchKMeans
from sklearn.model_selection import train_test_split

from scipy.sparse import issparse

from g_candidate.fit_encoder import fit_encoder
from g_candidate.coview_graph import CoViewGraph
from g_candidate.item.classificator import ItemClassificator
from g_candidate.item.preprocessor import ItemPreprocessor

max_cluster_size = 40

class DatasetUpdateCommandHandler():
    def __init__(self, graph: CoViewGraph, query_service, item_classificator:
ItemClassificator):
        self.__graph__ = graph
        self.__query_service__ = query_service
        self.__item_classificator__ = item_classificator

    def handle(self, command) -> None:
        dataset = pandas.read_csv(command.dataset_path)

        ids = dataset[command.id_column]
        data = dataset.drop(columns = command.ignore_columns + command.id_column)

        encoder = fit_encoder(data)

        encoded = encoder.transform(data)

        X = encoded.toarray()

        pca_type = SparsePCA if issparse(X) else PCA

        dimensity_reducer = pca_type(n_components = 2).fit(X)
        transformed_2d = dimensity_reducer.transform(X)

        n_clusters = round(X.shape[0] / max_cluster_size)

        k_means = MiniBatchKMeans(n_clusters = n_clusters, batch_size = 250)
        k_means.fit(transformed_2d)

        item_preprocessor = ItemPreprocessor(encoder, dimensity_reducer)
        self.__item_classificator__.update_model(k_means, item_preprocessor)

        clusters = k_means.predict(transformed_2d)

        for item_id, cluster in zip(ids, clusters):
            self.__graph__.add_item(item_id, cluster)
```

```
        sequences = self.__query_service__.fetch_queries()

        X = list(map(lambda seq: seq.Views, sequences.values()))
        Y = list(map(lambda seq: seq.PurchaseId, sequences.values()))

        for x, y in zip(X, Y):
            self.__graph__.register_purchase(y, x)

# додавання нового лоту
# new_items_handler.py

import pandas

from g_candidate.coview_graph import CoViewGraph
from g_candidate.item.classificator import ItemClassificator


class NewItemsCommandHandler():
    def __init__(self, graph: CoViewGraph, item_classificator:
ItemClassificator):
        self.__graph__ = graph
        self.__item_classificator__ = item_classificator

    def handle(self, command):
        items = pandas.DataFrame(command.items)

        ids = items[command.id_column]
        data = items.drop(columns = command.ignore_columns + command.id_column)

        clusters = self.__item_classificator__.predict(data)

        for item_id, cluster in zip(ids, clusters):
            self.__graph__.add_item(item_id, cluster)

# робота з графом суміжних переглядів
from neo4j import GraphDatabase

class CoViewGraph(object):
    def __init__(self, uri, user, password):
        self._driver = GraphDatabase.driver(uri, auth=(user, password))

    def close(self):
        self._driver.close()

    def add_item(self, item_id, cluster_id):
        with self._driver.session() as session:
            session.write_transaction(self._create_item, str(item_id),
str(cluster_id))
```

```
    def deactivate_item(self, item_id):
        with self._driver.session() as session:
            session.write_transaction(self._deactivate_item, str(item_id))

    def register_purchase(self, target_item, viewed_items):
        viewed_items = list(map(lambda i: str(i), viewed_items))
        with self._driver.session() as session:
            session.write_transaction(self._register_purchase, str(target_item),
viewed_items)

    def drop_clusters(self):
        with self._driver.session() as session:
            session.write_transaction(self._drop_clusters)

    def get_recommended_items(self, viewed_items, top_n):
        viewed_items = list(map(lambda i: str(i), viewed_items))
        with self._driver.session() as session:
            records = session.run(
                "MATCH (view_group:ItemsCluster)<-[:BelongsTo]-(view_item:Item) "
                "WHERE view_item.id IN $viewed_items "
                "WITH view_group AS vg "
                "MATCH (tg:ItemsCluster)<-[r:CoView]-(vg) "
                "RETURN { id: tg.id, weight: 1.0 - reduce(prod = 1.0, w IN
collect(r.weight)| prod * (1.0 – w)) } AS gw",
                viewed_items = viewed_items
            )

            records = list(map(lambda record: record['gw'], records))
            records.sort(key=lambda i: i['weight'], reverse=True)

            top_n = min(top_n, len(records))

            clusters = records[:top_n]
            clusters = list(map(lambda i: i['id'], clusters))

            records = session.run(
                "MATCH (c:ItemsCluster)<-[:BelongsTo]-(i:Item{ available: true })
"
                "WHERE c.id IN $clusters "
                "RETURN i.id",
                clusters=clusters
            )

            return list(map(lambda r: r['i.id'], records))

    @staticmethod
    def _create_item(tx, item_id, cluster_id):
        tx.run(
            "MERGE (i:Item { id: $item_id }) "
```

```
            "ON CREATE SET i.available = true "
            "WITH i "
            "MERGE (c:ItemsCluster { id: $cluster_id }) "
            "WITH c, i "
            "MERGE (c)<-[r:BelongsTo]-(i) "
            "RETURN c, r, i",
            item_id=item_id,
            cluster_id=cluster_id
        )

    @staticmethod
    def _deactivate_item(tx, item_id):
        tx.run(
            "MATCH (i:Item { id: $item_id, available: true }) "
            "SET i.available = false",
            item_id=item_id
        )

    @staticmethod
    def _drop_clusters(tx):
        tx.run("MATCH (a:ItemsCluster)<-[r]-(b) DELETE a, r")

    @staticmethod
    def _register_purchase(tx, target_item, viewed_items):
        tx.run(
            "MATCH (target_group:ItemsCluster)<-[:BelongsTo]-(:Item { id:
$target_item_id }) "
            "WITH target_group AS tg "
            "MATCH (view_group:ItemsCluster)<-[:BelongsTo]-(view_item:Item) "
            "WHERE view_item.id IN $viewed_item_ids "
            "WITH collect(view_group.id) AS viewed_groups, tg "
            "MATCH (g:ItemsCluster) "
            "WHERE g.id IN viewed_groups "
            "WITH viewed_groups, g, tg "
            "MERGE (tg)<-[r:CoView]-(g) "
            "ON CREATE SET r.weight = 0.0 "
            "WITH viewed_groups, g, tg "
            "MATCH (tg)<-[r:CoView]-(g:ItemsCluster) "
            "WHERE g.id IN viewed_groups OR r.weight > 0.0  "
            "SET r.weight = ( "
            "   CASE "
            "    WHEN g.id IN viewed_groups "
            "       THEN (r.weight + 1) * 1.0 / 2 "
            "   ELSE r.weight * 1.0 / 2 "
            "    END "
            ")",
            target_item_id = target_item,
            viewed_item_ids = viewed_items
        )
```

```
# оцінка моделі
# eval_model.py

import pandas
import random
import datetime
from collections import namedtuple
from sklearn.model_selection import train_test_split
from coview_graph import CoViewGraph

df = pandas.read_csv('events.csv')

df.sort_values(by=['timestamp'], ascending=[False])

items = df['itemid'].unique()

data = list()

CoViewEntry = namedtuple('CoViewEntry', [ 'SequenceId', 'PurchaseId', 'ViewId',
'Time' ])

total_count = df.size
current_position = 0

purchases_count = 0

ver = datetime.datetime.now().timestamp()

for index, row in df.iterrows():
    current_position += 1
    print('%d / %d Processing ..' % (current_position, total_count))

    if row['event'] != 'view':
        purchases_count += 1

        user = row['visitorid']
        to_time = row['timestamp']

        from_time = datetime.datetime.now() - datetime.timedelta(days=3)

        coviews = df.loc[(df['event'] == 'view') & (df['timestamp'] < to_time) &
(df['timestamp'] > from_time.timestamp()) & (df['visitorid'] == user)]

        coviews.drop_duplicates(subset='itemid', keep='last', inplace=True)

        for _, coview in coviews.iterrows():
            data.append((CoViewEntry(SequenceId=index, PurchaseId=row['itemid'],
ViewId=coview['itemid'], Time=coview['timestamp'])))
```

```
    print('%d / %d Done' % (current_position, total_count))

    if purchases_count == 10000:
        break

print('Saving data to csv ..')
df = pandas.DataFrame(data, columns =[ 'SequenceId', 'PurchaseId', 'ViewId',
'Time' ])
df.to_csv('data_%s.csv' % ( ver ))
print('Saving data to csv Done')

CoViewSequence = namedtuple('CoViewSequence', [ 'PurchaseId', 'Views' ])

print('Prepearing sequences ..')
sequences = dict()

for _, row in df.iterrows():
    seq = sequences.get(row['SequenceId'],
CoViewSequence(PurchaseId=row['PurchaseId'], Views=list()))
    seq.Views.append(row['ViewId'])
    sequences[row['SequenceId']] = seq

print('Prepearing sequences Done')
print('Spliting dataset ..')
X = list(map(lambda seq: seq.Views, sequences.values()))
Y = list(map(lambda seq: seq.PurchaseId, sequences.values()))

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1,
random_state = 0)

print('Spliting dataset Done')


graph = CoViewGraph('bolt://localhost:7687', 'adm', 'qwerty')

items = list(set(df['PurchaseId'].tolist() + df['ViewId'].tolist()))

position = 0
total = len(items)

print('Adding items to graph ..')

for item in items:
    position += 1
    graph.add_item(item, item)
    print('%d / %d Done' % (position, total))

position = 0
```

```
total = len(X_train)

print('Registering purchases ..')

for x, y in zip(X_train, Y_train):
    graph.register_purchase(y, x)
    position += 1
    print('%d / %d Done' % (position, total))

predicted_count = 0
succes_count = 0
total = len(X_test)

for x, y in zip(X_test, Y_test):
    predicted = graph.get_recommended_items(x, 50)

    predicted_count += 1

    succes_count += 1 if str(y) in predicted else 0

    acc = succes_count / predicted_count

    print('%d / %d Done; success: %d' % (predicted_count, total, succes_count))

    graph.register_purchase(y, x)
```